SUNMI PAY SDK V2 Development Document

Shanghai Sunmi Tech Co.,Ltd

3/F, 7 KIC, 388 Songhu Rd, Yangpu District, Shanghai，China

目录

1. Revision History

Doc Version	Release	Changes	The appropriate SDK version	
3.0.0	2017/11/30	1st Draft	SunmiPaySDKService1.0.1	
3.0.1	2018/1/22	Revise Docs	SunmiPaySDKService1.0.3/SunmiPayHardwareService3.0.44	
3.0.2	2018/2/23	Add Error Code	SunmiPaySDKService1.0.4/SunmiPayHardwareService3.0.44	
3.0.3	2018/3/21	Add setSysParam(), Add TransPreProcess()	SunmiPaySDKService1.0.6/SunmiPayHardwareService3.1.2	
3.0.4	2018/4/12	Add smartCardExchange()parameter definition.	SunmiPaySDKService1.0.6/SunmiPayHardwareService3.1.2	
3.0.5	2018/5/3	Support SM4 algorithm.	SunmiPayHardwareService_v3.2.07	
3.0.6	2018/5/18	Add the SysParam constant definition of the keyword "RESERVED" on and off the buzzer instructions.	SunmiPayHardwareService_v3.2.07	
3.0.7	2018/6/5	Add mifareReadBlock() blockData format definition	SunmiPayHardwareService_v3.2.07	
3.0.8	2018/8/16	Support DUKPT.	SunmiPayHardwareService_v3.2.18	
3.0.9	2018/8/31	Add detectCard() and read card info	SunmiPayHardwareService_v3.2.19	
3.1.0	2018/9/21	Modify the description for apduCommand()	SunmiPayHardwareService_v3.2.19	
3.2.0	2018/10/12	Support V2 api.	SunmiPayHardwareService_v3.2.20	
3.2.1	2018/10/22	Add error code	SunmiPayHardwareService_v3.2.20	
3.2.2	2018/10/24	Add API: smartCardExchange() and smartCardExchangeNISO()	SunmiPayHardwareService_v3.3.0	
3.2.3	2018/11/01	1.Fix issue: wrong params order of dataEncrypt () 2.Modify method comments in SunmiPayKernel	SunmiPayHardwareService_v3.3.0	
3.2.4	2018/11/02	Modify Class EMVTransDataV2	SunmiPayHardwareService_v3.3.0	
3.2.5	2018/11/05	1.add API: initEmvProcess() 2.Modify get/Set TLV data API 3.Modify EMV flow chart	SunmiPayHardwareService_v3.3.0	
3.2.6	2018/11/07	1.Delete fields of **EMVTransDataV2** 2.Update EMV flow chart	SunmiPayHardwareService_v3.3.0	
3.2.7	2018/11/15	1.Update describe of PinPadConfigV2 2.Modify the getKeyCheckValue description 3.EMV procedure add interfaces: onAppFinalSelect(),importAppFinalSelectStatus(). 4.remove interface	SunmiPayHardwareService_v3.3.0	

		apduCommandNISO()		
3.2.8	2018/11/16	1.change name smartCardExChangeNISO()to transmitApdu() 2.transmitApdu() return card data directly(remove fields len,swa,swb)	SunmiPayHardwareService_v3.3.0	
3.2.9	2018/11/19	Modify interface transmitApduset, change max size of sendBuff, recvBuff to 255B	SunmiPayHardwareService_v3.3.0	
3.2.10	2018/11/22	Change type of **certType** to **int** in interface EMVListenerV2. onCertVerify()	SunmiPayHardwareService_v3.3.0	
3.2.11	2018/11/24	1.add interface EMVListenerV2. onRequestSignature(),importSignatureStatus() 2.add error code -50027(Signature error) 3.Modify interface buzzerOnDevice()	SunmiPayHardwareService_v3.3.0	
3.2.12	2018/11/29	1.add interface EMVListenerV2.onCardDataExchangeComplete()	SunmiPayHardwareService_v3.3.0	
3.2.13	2018/12/04	1.add interface EMVListenerV2. onConfirmationCodeVerified() 2.add error code -50028 3.update emv flow chart	SunmiPayHardwareService_v3.3.0	
3.2.14	2018/12/08	1.update interface EMVListenerV2.onWaitAppSelect(),change paramter type to List<EMVCandidateV2> 2.modify interface EMVOptV2.importOnlineProcStatus()	SunmiPayHardwareService_v3.3.0	
3.2.15	2018/12/12	1.Add code depiction of callback EMVListenerV2.onTransResult() 2.Modify depiction of parameter **status** in interface **importOnlineProcStatus**()	SunmiPayHardwareService_v3.3.0	
3.2.16	2018/12/13	Increase -20001——-20006 error code	SunmiPayHardwareService_v3.3.0	
3.2.17	2019/01/07	Add RSA interfaces	SunmiPayHardwareService_v3.3.16	
3.2.18	2019/01/17	1.Add interface removeRSAKey() 2.Remove the T=0,T=1 description i-n interface transmitApdu()	SunmiPayHardwareService_v3.3.17	
3.2.19	2019/01/21	Update the keyIndex params illustr-ation of RSA/Cetificate related interfaces	SunmiPayHardwareService_v3.3.17	
3.2.20	2019/03/18	1.Update interface saveKeyDukpt(), add code to generate initialize KSN 2.Add interface dukptGetInitKsn()	SunmiPayHardwareService_v3.3.28	

		3.Update interface transmitApdu(),support contact card. 4.Update interface apduCommand(), modify fields Lc,Le,outLen value range of param ApduSendV2, ApduRecvV2		
3.2.21	2019/03/19	1.Update interface saveKeyDukpt(),remove generate initialize ksn code 2.Update interface dukptGetInitKSN() 3.Add interface EMVOptV2.abortTransactProcess()	SunmiPayHardwareService_v3.3.29	
3.2.22	2019/04/10	1.add MifarePlus SL3 card data exchange interfaces 2.add SLE4442/4428 card data exchange interfaces 3.add AT24C01/02/04/08/16/32/64/128/256/512 card data exchange Interfaces 4.add embedded doc [**Sunmi PinBlock format**]	SunmiPayHardwareService_v3.3.32	
3.2.23	2019/05/29	1.add interface EMVOptV2.transactProcessEx(),EMVOptV2.importDataExchangeStatus() 2.add interface EMVListenerV2.onRequestDataExchange()	SunmiPayHardwareService_v3.3.40	
3.2.24	2019/07/06	1.add at88scxx card data exchange interfaces 2.add interface transmitApduEx(),support customized Miafre card APDU data exchange 3.remove firmware version of revision history column [The appropriate SDK version] 4.Add interface sysGetRandom() 5.Add RSA signing/verify signature interaces	SunmiPayHardwareService_v3.3.46	
3.2.25	2019/07/30	1.Add key inject interfaces injectPlaintextKey(),injectCiphertextKey() 2.CheckCardCallbackV2 add callback methods findICCardEx(),findRFCardEx(),onErrorEx() 3.Modify CheckCardCallbackV2. findMagCard(), add key cardType, trackErrorCode,etc in param info. 4.Add interface EMVOptV2.queryECBalance()	SunmiPayHardwareService_v3.3.48	

3.3.26	2019/09/16	1.Modify illustration of interface checkCard() 2.AidV2 Add fields clsStatusCheck, zeroCheck 3.EMVOptV2 add interfaces addDrlLimit Set(), deleteDrlLimitSet(), setTer mParamEx() to support DRL	SunmiPayHardwareService_v3.3.52	
3.2.27	2019/10/15	1.Add system parameter constants: EM VVersion,PaypassVersion,PaywaveVe rsion,QPBOCVersion,EntryVersion,M irVersion,JCBVersion,PAGOVersion, EmvKernelCheckSum 2.Change the hit text of input PIN 3.Add Error code -70001~ -70006	SunmiPayHardwareService_v3.3.53	
3.2.28	2019/10/28	1.EMVOptV2 add interface queryAidCapk Lis()、transactPreProcess() 2.PinPadOptv2 add interface cancelInp utPin() 3. add cardCategory, atqa in radio ca rd check card call back.	SunmiPayHardwareService_v3.3.55	
3.2.29	2019/11/29	1. Security module add interfaces dat aEncryptDukptEx(),dataDecryptDukptEx (),calcMacDukptEx(),verifyMacDukptEx (),saveTR31Key(),saveCiphertextKeyRS A(),saveRSAKey() 2.Add system Param key：AEVersion 3.Add new EMV flowType, value is 0x04 4.EMV module add interfaces addRevocList()、deleteRevocList()、 sysSetTime()、sysGetTime()、clearData () 5.Modify param name in interfaces sig ningRSA(),verifySignatureRSA()	SunmiPayHardwareService_v3.3.58	
3.2.30	2020/02/19	1.AidV2 add fields kernelType,paramTy pe 2.Add system parameter constants:FLAS HVersion 3.Change constant names in EMV FlowTy pe	SunmiPayHardwareService_v3.3.62	
3.2.31	2020/02/24	1.Card module add interfaces: ctx512R eadBlock(),ctx512WriteBlock(),ctx512 UpdateBlock(),ctx512GetSignature(),c tx512MultiReadBlock(),mifareIncValue Dx(),mifareDecValueDx(),mifareTransf er(),mifareRestore()	SunmiPayHardwareService_v3.3.64	

		2.PinPad moudle add interface: setPinPadText() 3.Security moudle add interface: deleteKey(),saveKeyDukptAES() 4.SysParam add keyword:PCD_PARAM_A,PCD_PARAM_B,PCD_PARAM_C		
3.2.32	2020/04/22	1.Card moudle add interfaces checkCardEx(),transmitApduExx(),transmitMultiApdus() 2.Basic module add interfaces setStatusBarDropDownMode(),setNavigationBarVisibility(),setHideNavigationBarItems(),sysPowerManage() 3.Add system param constants: DPASVersion、APEMVVersion 4.Add EMV transaction result code constants:AidlConstants.EMV.TransResult	SunmiPayHardwareService_v3.3.66	
3.2.33	2020/05/22	1. Modify illustration of interface setTermParamEx()	SunmiPayHardwareService_v3.3.74	
3.2.34	2020/06/02	1.PinPad moudle add interfaces setPinPadMode(), getPinPad()	SunmiPayHardwareService_v3.3.76	
3.2.35	2020/06/04	1. Modify illustration of interface setTermParamEx()	SunmiPayHardwareService_v3.3.77	
3.2.36	2020/06/16	1.Card moudle add interface checkCardEnc() 2.Add key “pan”,”name”,”expire” in CheckCardCallbackV2.findMagCard()	SunmiPayHardwareService_v3.3.80	
3.2.37	2020/06/17	1.Change doc veriosn to v3.2.37	SunmiPayHardwareService_v3.3.81	
3.2.38	2020/07/03	--	--	
3.2.39	2020/07/06	1.Modify illustration of interface EMV transactProcessEx() 2.Modify illustraction of interface checkCardEx()	SunmiPayHardwareService_v3.3.86	
3.2.40	2020/09/11	1. SecuityOptV2 add interfaces: calcMacEx(),generateSM2Keypair(),injectSM2Key(),sm2Sign(),sm2VerifySign()、sm2EncryptData()、sm2DecryptData()、calcSecHash() 2. EMVOptV2 add interfaces: setAccountDataSecParam(),getAccountSecData() 3. PinPadOptV2 add interface: initPinPadEx() 4. Add system param constants： ”PUREVersionFul”,”EFTPOSVersionFull”	SunmiPayHardwareService_v3.3.95	

		,”APEMVVersionFull” 5.change dukpt-3DES key index range to 0-9 or 1100-1199 6.Add dataIn length limit when dukpt keyIndex in 1100-1199		
3.2.41	2020/11/06	1.Add key index illustration in Appendix	SunmiPayHardwareService_v3.3.95	
3.3.42	2020/11/11	1.Add RSA transformation: RSA/ECB/OAEPWithSHA-1AndMGF1Padding, RSA/ECB/OAEPWithSHA-256AndMGF1Padding, RSA/ECB/OAEPWithSHA-512AndMGF1Padding	SunmiPayHardwareService_v3.3.96	
3.2.43	2020/12/16	1. EMVOptV2.setTermParamEx() add key contactlessManualSelApp, importScriptData 2.AidlConstants.EMV.TLVOpCode add constant value OP_ADD_SELF_DEFINE_TAG, OP_DEL_SELF_DEFINE_TAG 3.SystemParam add key “SecMode”,” PCD_IFMVersion”	SunmiPayHardwareService_v3.3.98	
3.2.44	2020/12/29	1.update interface saveKeyDukptAES(), set param ksn length as 12	SunmiPayHardwareService_v3.3.99	
3.2.46	2021/02/04	1. EMVOptV2 add interface importTermRiskManagementStatus() 2. EMVListenerV2 add interface onTermRiskManagement()	SunmiPayHardwareService_v3.3.102	
3.2.47	2021/04/28	1.ReadCardOptV2 add smartCardIoControl() interface 2.SecurityOptV2 add verfiryMac()、RSA、RKI related interfaces 3.update SecurityOptV2 MKSK/Dukpt key index 4.remove keyword Sysparam.SDK_VERSION	SunmiPayHardwareService_v3.3.112	
3.2.48	2021/07/27	1.PinPadOptV2 add interface setAntiExhaustiveProtectionMode(),getAntiExhaustiveProtectionMode()	SunmiPayHardwareService_v3.3.126	
3.2.49	2021/08/30	1.update Error code, add libbase error code	SunmiPayHardwareService_v3.3.136	
3.2.50	2021/09/16	1. add constant AidlConstants.EMV.KernelType.RUPAY 2.Update comment of interface dukptIncreaseKSN ()	SunmiPayHardwareService_v3.3.137	

2. Abstract

2.1 Introduction

SunmiPaySDK is a set of interface which is based on firmware encapsulation and is close to Java developer to call hardware. Through this SDK, developer can quickly call the corresponding firmware interface of Sunmi financial instruments and realize the development of its own business logic. SDK mainly includes: terminal information basic module, card operation module, password keyboard module, EMV module, security module.
This document is the SunmiPaySDK V2 interface documentation. Compared to the V1 interface, the V2 interface makes it easier to develop this understanding and call.

2.2 SDK Integration

2.2.1 Quick integration

This document is an integrated development guide for the SUNMI’ s POS. To guide the use method of SDK, It requires the reader to have already familiar with the basic use method of IDE, have certain Android programming basis, familiar with financial related specifications, processes and related concepts (key, pinblock, pan,emv, MAC, etc.). At present, SDK only supports the development of Sunmi P1N, P1-4G, P2 Pro, P2 Lite. Please read this document carefully before using SDK. Check the following prerequisites before use.

1. Model number is P1N or P1-4G or P2 Pro or P2 Lite.(Setting-About device-Model number)
2. Go to Setting—>APP—>top right cornor—>show system SunmiPayHardwareService_v3.3.0_release.apk or the latest version.
3. Android studio quick integrate，Put the PayLib-release-xxx.aar package under the libs folder.

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

Build the Project again when finished.

2.2.2 Android version and IDE version supported by the SDK

SDK only support API-19(Android 4.4) or the latest version.
SDK only support Android studio、Intellij.

3. API

3.1 SunmiPayKernel SDK Operation Object

3.1.1 Get an instance of SunmiPayKernel

Prototype	SunmiPayKernel getInstance()	
Feature	Get kernel instance object of SDK	
Parameter	[in]	None
Return	SunmiPayKernel	
Comment	None	

3.1.2 Connect to PaySDK

Prototype	initPaySDK (Context context,ConnectCallback connectCallback)	
Feature	Initialize Pay SDK.	
Parameter	context[in]	Context
	connectCallback[in]	Callback of connection status. Refer to ConnectCallback
Return	None	
Comment	It is recommended to call this method in your app's Application class, if failure to call this method, any invoking of SDK API method may throw NullPointerException.	

3.1.3 Disconnect to PaySDK

Prototype	destroyPaySDK ()	
Feature	Destroy the Pay SDK.	
Parameter	[in]	None
Return	None	
Comment	This method will disconnect your app to Pay SDK. After calling this method, any function module in SunmiPayKernel will be set to **null**, and any invoke of SDK API method will throw NullPointerException. To prevent a memory leak, call this methos in Activity's onDestroy() is best practice.	

3.1.4 ConnectCallback

3.1.4.1 Connect to PaySDK callback

Prototype	onConnectPaySDK ()	
Feature	Callback method, indicate connect to PaySDK success	
Parameter	[in]	None
Return	None	
Comment	Any SDK API method calls must be made after receiving this callback. After initializing the SDK, developers should pay attention to this method. When this method called, it indicates the SDK was initialized successfully, and the SDK related API can be invoked rightly.	

3.1.4.2 Disconnect to PaySDK callback

Prototype	onDisconnectPaySDK ()	
Feature	Disconnect the PaySDK	
Parameter	[in]	None
Return	None	
Comment	When this method called, it indicates your app has disconnected to Pay SDK, any invoke of SDK API method may produce Exception. To use SDK API, please call method InitPaySDK() firstly.	

3.2 Public member variable

After connected to PaySDK, in SunmiPayKernel, there are some function modules can be used to interactive with PaySDK, as following table showing:

Variable name	Description	Comment
mPinPadOptV2	Pinpad operation module	Includes PinPad API
mBasicOptV2	Basic operation module	Get basic info , control LED , control Buzzer.
mReadCardOptV2	Read card module	Read card
mEMVOptV2	EMV module	Emv operation API
mSecurityOptV2	Security module	Save key , mac calculation , encrypt data

3.3 Basic operation module

3.3.1 Get system parameter

Prototype	String getSysParam(String key)	
Feature	Read the properties of the system resource key through the user parameter keyword.	
Parameter	key[in]	Refer to AidlConstantsV2. SysParam
Return	Attribute values queried	
Comment	If the required attribute value does not exist, then return "NULL".	

3.3.2 Set system parameter

Prototype	int setSysParam(String key, String value)	
Feature	Set the properties of the system resource key through the user parameter keyword.	
Parameter	key[in]	Refer to AidlConstantsV2. SysParam (only support keyword: RESERVED)
	value[in]	Keyword’ s value
Return	0: Success Other value: Fail	
Comment	Only support keyword: RESERVED	

3.3.3 Control Buzzer

Prototype	void buzzerOnDevice(int count, int freq, int duration, int interval)	
Feature	Control the Buzzer on the device.	
Parameter	count[in]	Beep times (0~100)
	freq[in]	The frequency of tone (unit: Hz)
	duration[in]	The duration time (unit: ms)
	interval[in]	The time interval between two beeps (unit: ms) 0~10000
Return	None	
Comment	On device P2lite/P2Pro/P2, there are no really buzzer exist, the beep sound made by loudspeaker, and param **freq** and **duration** are invalid	

3.3.4 Control Led

Prototype	int ledStatusOnDevice (int ledIndex, int ledStatus)	
Feature	Control Led status on device	
Parameter	ledIndex[in]	Refer to LedLight definition: AidlConstantsV2.LedLight
	ledStatus[in]	LED status, 1-LED off; 0-LED on.

Return	0: Success Other value: Fail	
Comment	None	

3.3.5 Set screen mode

Prototype	int setScreenMode(int mode)	
Feature	Set the screen exclusive, disable the bottom navigation bar and the SystemUI drop-down box, and disable the volume key.	
Parameter	mode[in]	Set screen exclusive mode.1：Set screen exclusive，-1: Set screen exclusive exit.
Return	0: Success Other value: Fail	
Comment	Setting up screen exclusivity usually needs an appropriate time to cancel. It is recommended that the user use this interface with the power management lock to keep the screen always bright and unlocked. Otherwise, the screen exclusivity can be maintained after the information screen appears. Only by unplugging the battery and restarting the device can the screen exclusivity be removed.	

3.3.6 Get random data

Prototype	int sysGetRandom(byte[] randData, int len)	
Feature	Get a specific length random data from SDK	
Parameter	randData[out]	Buffer, store the random data
	len[in]	The data length, range 0~256
Return	0: Success Other value: Fail	
Comment	None	

3.3.7 Control Led

Prototype	int ledStatusOnDeviceEx(int redStatus, int greenStatus, int yellowStatus, int blueStatus)	
Feature	Control all Leds status on device at one time	
Parameter	redStatus[in]	Red LED status, 0-on, 1-off
	greenStatus[in]	Green LED status, 0-on, 1-off
	yellowStatus[in]	yellow LED status, 0-on, 1-off
	blueStatus[in]	Blue LED status, 0-on, 1-off 6
Return	0: Success Other value: Fail	
Comment	None	

3.3.8 Set status bar dropdown mode

Prototype	int setStatusBarDropDownMode(int mode);	
Feature	Set status bar dropdown mode	
Parameter	mode[in]	0-enable dropdown, 1-disable dropdown
Return	0: Success Other value: Fail	
Comment	None	

3.3.9 Set navigation bar visibility

Prototype	int setNavigationBarVisibility(int visibility)	
Feature	Set navigation bar visibility	
Parameter	visibility[in]	Navigation bar visibility, 0-Gone, 1-Visible
Return	0: Success Other value: Fail	
Comment	None	

3.3.10 Set hide navigation bar items

Prototype	int setHideNavigationBarItems(int flag)	
Feature	Hide navigation bar items	

Parameter	flag[out]	Composite value, the items to be hide:
		STATUS_BAR_DISABLE_HOME=0x00200000;//hide home key
		STATUS_BAR_DISABLE_BACK = 0x00400000;//hide back key
		STATUS_BAR_DISABLE_RECENT = 0x01000000;//hide recent key
		Set **flag=STATUS_BAR_DISABLE_HOME | STATUS_BAR_DISABLE_BACK**
		to hide home and back key

Return	0: Success Other value: Fail	
Comment	None	

3.3.11 Manage device power

Prototype	int sysPowerManage(int mode)	
Feature	Manage deivce power	
Parameter	mode[in]	Device power mode, 1-dormant(not support), 2-shutdown, 3-reboot

Return	0: Success Other value: Fail
Comment	None

3.4 Card operation module

3.4.1 ReadCardOpt

3.4.1.1 Check card

Prototype	void checkCard(int cardType, CheckCardCallbackV2 callback, int timeout)	
Feature	For card checking, magnetic stripe card, IC card and NFC card are supported. After checking, the card type will be put into CheckCardCallbackV2.	
Parameter	cardType[in]	Composite card types, support checking multi cards. This param can be a composite value of CardType.getValue(), for example, check MAGNETIC,NFC,IC at the same time, set this param as: CardType.MAGNETIC.getValue()|CardType.NFC.getValue()| CardType.IC.getValue()
	callback[in]	Check card callback. Refer to CheckCardCallbackV2
	timeout[in]	Timeout (unit: second). Effective time range: 1-120s
Return	None	
Comment	No distinction between bank card and non bank card.	

3.4.1.2 Cancel check card

Interface instructions: Artificial return must call cancelCheckCard(), terminate the underlying blocking thread, otherwise the next execution function will fail (for example, click the physical return key, click the interface navigation bar return key to call the function)

Prototype	void cancelCheckCard();	
Feature	Cancel check card	
Parameter	[in]	None
Return	None	
Comment	The function needs to be called when the CheckCard is not returned (CheckCardCallbackV2 Success or Failed Interface is not callback) and leaves the interface.	

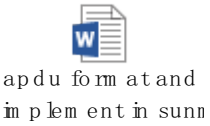
3.4.1.3 APDU command exchange

Prototype	int apduCommand (int cardType, ApduSendV2 send, ApduRecvV2 recv)
Feature	IC card operation function. This function support the general interface protocol for IC

	card.(T=0 and T=1) This support the general interface protocol for contactless card. (T=CL)	
Parameter	cardTypy[in]	Card type currently in operation

	send[in]	class ApduSendV2 {
		byte[] command; // Command[] = {CLA , INS , P1 , P2}.
		short lc; // Length of dataIn (0~256)
		byte[] dataIn; // Data of the IC card which you want to send, max length is 256 bytes.
		short le; // Expected length of the returned data (0~256)
		}

	recv[out]	class ApduRecvV2 {
		short outLen; //Actual data length returned from IC card (0~256)
		byte[] outData; // Data returned from IC card. max length is 256 bytes.
		byte swa; // Status byte 1
		byte swb; // Status byte 2
		}

Return	0: Success Other value: Fail	
Comment	For variable format of ApduSendV2 , refer to the doc **apdu format and implement in sunmi way** 	

3.4.1.4 APDU command exchange (not recommend)

Prototype	int smartCardExchange (int cardType, byte[] apduend, byte[] apduRecv)	
Feature	IC card operation function. This function support the general interface protocol for IC card.(T=0 and T=1) This support the general interface protocol for contactless card. (T=CL)	
Parameter	cardTypy[in]	Card type currently in operation
	apduSend[in]	The APDU command which to send, format is: Command(4B)+LC(1B, value is **len**)+inData(**len** B)+LE(1B) LC is unsigned, range is 0~255
	apduRecv[out]	The out data buffer, apduRecv.length>=260 , receive data format is : outLen(2B, Big endian,value is **len**)+outData(**len** B)+SWA(1B)+SWB(1B)
Return	0: Success Other value: Fail	
Comment	None	

3.4.1.5 Transmit APDU command to card

Prototype	int transmitApdu (int cardType, byte[] sendBuff, byte[] recvBuff)	
Feature	Transmit the **sendBuff** to card directly. This function support the general interface protocol for IC card.(T=0 and T=1), and also support the general interface protocol for contactless card. (T=CL)	
Parameter	cardType[in]	Card type currently in operation
	sendBuff[in]	Data which want to transmit to card. Max length is 255B
	recvBuff [out]	Data received from card. The max receive data length is 255B, recvBuff.lenght>=255
Return	>=0: The valid data length in recvBuff <0: Error code	
Comment	None	

3.4.1.6 Close contact and contactless module

Prototype	int cardOff(int cardType)	
Feature	Power off contact IC card or turn off carrier of contactless card.	
Parameter	cardType[in]	Card type, support multiple card for checking card. Refer to: AidlConstantsV2.CardType.getValue().Fill in the card type according to the card type in checkCard, one at a time.
Return	0: Success Other value: Fail	
Comment	None	

3.4.1.7 Check if card exist on slot

Prototype	int getCardExistStatus(int cardType)	
Feature	Check if card exist on slot	
Parameter	cardType[in]	Card type (non-composite type), only supports: NFC、IC、PSAM0、PSAM1 Can only be one of 4 types at a time.

Return	AidlConstantsV2.CardExistStatus
	CARD_ABSENT = 0x01; // Card don't exist.
	CARD_PRESENT = 0x02; // Card exist.	

Comment		

3.4.1.8 Mifare Classic

3.4.1.8.1 Verify password of M1 card sector

Prototype	int mifareAuth(int keyType, int block, byte[] key)	
Feature	Submit password A or B of responding read-write block when verifying the access of M1 card.	
Parameter	keyType[in]	Specified password type: 0 --- submit password A 1 --- submit password B
	block[in]	Block number for specified access. The available range is 0~63 for M1 card with 1K.
	key[in]	Point to submitted password buffer(6 Bytes)
Return	0: Success Other value: Fail	
Comment	None	

3.4.1.8.2 M1 card read block data

Prototype	int mifareReadBlock(int block, byte[] outData)	
Feature	Read the contents in the specified block of M1 card. (16 bytes)	
Parameter	block[in]	Block number for specified access. The available range is 0~63 for M1 card with 1K.
	outData [out]	Block data.
Return	>=0: The valid data length of **outData** <0: Error code	
Comment	None	

3.4.1.8.3 M1 card write block data

Prototype	int mifareWriteBlock(int block, byte[] data)	
Feature	Write specified content in specified block for M1 card (16 bytes).	
Parameter	block[in]	Block number for specified access. The available range is 0~63 for M1 card with 1K.
	data[in]	Block data.
Return	0: Success Other value: Fail	
Comment	None	

3.4.1.8.4 M1 card operate sector data (Increment operation)

Prototype	int mifareIncValue(int block, byte[] value)	
Feature	Charge for wallet of M1 card.	
Parameter	block[in]	Use to assign the number of block where the wallet is in it. The available range is 0~63 for M1 card with 1K.
	value[in]	Point to first address of buffer for the amount of charging. Totally 4 bytes for the amount and the lower byte is at the beginning.
Return	0: Success Other value: Fail	
Comment	None	
Example		

3.4.1.8.5 M1 card operate sector data (decrement operation)

Prototype	int mifareDecValue(int block, byte[] value)	
Feature	Charge back for wallet of M1 card.	
Parameter	block[in]	Use to assign the number of block where the wallet is in it. The available range is 0~63 for M1 card with 1K.
	value[in]	Point to first address of buffer for the amount of charging back. Totally 4 bytes for the amount and the lower byte is at the beginning.
Return	0: Success Other value: Fail	
Comment	None	
Example		

3.4.1.8.6 M1 card operate sector data (Increment operation, shrink method)

Prototype	int mifareIncValueDx(int block, byte[] value)	
Feature	Charge for wallet of M1 card, not contains transfer operation.	
Parameter	block[in]	Use to assign the number of block where the wallet is in it. The available range is 0~63 for M1 card with 1K.
	value[in]	Point to first address of buffer for the amount of charging. Totally 4 bytes for the amount and the lower byte is at the beginning.
Return	0: Success Other value: Fail	
Comment	M1 card charge for wallet contains 2 steps: 1.Charge, and cache the value data in data register 2.Transfer the cached value to block Interface **mifareIncValue()** contains the 2 steps, **mifareIncValueDx()** contains step 1 only.	
Example		

3.4.1.8.7 M1 card operate sector data (decrement operation, shrink method)

Prototype	int mifareDecValueDx(int block, byte[] value)	
Feature	Charge back for wallet of M1 card, not contains transfer operation.	
Parameter	block[in]	Use to assign the number of block where the wallet is in it. The available range is 0~63 for M1 card with 1K.
	value[in]	Point to first address of buffer for the amount of charging back. Totally 4 bytes for the amount and the lower byte is at the beginning.
Return	0: Success Other value: Fail	
Comment	M1 card charge for wallet contains 2 steps: 1.Charge, and cache the value data in data register 2.Transfer the cached value to block Interface **mifareDecValue()** contains the 2 steps, **mifareDecValueDx()** contains step 1 only.	
Example		

3.4.1.8.8 M1 card transfer data to block

Prototype	int mifareTransfer(int destBlock)	
Feature	Transfer data register data to block	
Parameter	destBlock[in]	Block number
Return	0: Success Other value: Fail	
Comment	None	
Example		

3.4.1.8.9 M1 card restore block data

Prototype	int mifareRestore(int srcBlock)	
Feature	Restore block data to data register	
Parameter	srcBlock[in]	Block number
Return	0: Success Other value: Fail	
Comment	None	
Example		

3.4.1.9 Mifare Ultralight C

3.4.1.9.1 Verify Password of Mifare Ultralight C Card Sector

Prototype	int mifareUltralightCAuth(byte[] authKey)
Feature	Submit password A or B of responding read-write block when verifying the access of Mifare

	ultralight c.	
Parameter	authKey[in]	Point to submitted password buffer
Return	0: Success Other value: Fail	
Comment	None	

3.4.1.9.2 Mifare Ultralight C Read Block Data

Prototype	int mifareUltralightCReadData(int block,byte[] outData)	
Feature	Read the contents in the specified block of Mifare ultralight c	
Parameter	block[in]	Block number
	outData[out]	Block data.
Return	>=0: The valid data length of **outData** <0: Error code	
Comment	None	

3.4.1.9.3 Mifare Ultralight C Write Block Data

Prototype	int mifareUltralightCWriteData(int block,byte[] data)	
Feature	Write specified content in specified block for Mifare ultralight c.	
Parameter	block[in]	Block number
	data[in]	Block data.
Return	0: Success Other value: Fail	
Comment	None	

3.4.1.10 Mifare Plus SL3

3.4.1.10.1 Mifare Plus read block data

Prototype	int mifarePlusReadBlock(int block, byte[] key, byte[] outData)	
Feature	Mifare plus read block data.	
Parameter	block[in]	Block number for specified access. The available range is depends on card, eg: MF1PLUS80x contains 32 4block sectoers and 8 16block secoters, each block size is 16 bytes, total memory is 32*4*16+8*16*16=4096, total block block count is 32*4+8*16=256, block range is 00~FF.
	key[in]	The block key, 16 bytes.
	outData[out]	Buffer, store the block data, **outData**.lenght>=16
Return	>=0: The valid data length of **outData** <0: Error code	

Comment	None
Example	None

3.4.1.10.2 Mifare Plus write block data

Prototype	int mifarePlusWriteBlock(int block, byte[] key, byte[] data)	
Feature	Mifare plus write block data.	
Parameter	block[in]	Block number for specified access. The available range is depends on card, eg: MF1PLUS80x contains 32 4block sectoers and 8 16block secoters, each block size is 16 bytes, total memory is 32*4*16+8*16*16=4096, total block block count is 32*4+8*16=256, block range is 00~FF.
	key[in]	The block key, 16 bytes.
	data[in]	The data to be write, 16 bytes
Return	0: Success Other value: Error code	
Comment	None	
Example	None	

3.4.1.10.3 Mifare Plus change block key

Prototype	int mifarePlusChangeBlockKey(int block, byte[] oldKey, byte[] newKey)	
Feature	Mifare plus change block key	
Parameter	block[in]	Block number for specified access. The available range is depends on card, eg: MF1PLUS80x contains 32 4block sectoers and 8 16block secoters, each block size is 16 bytes, total memory is 32*4*16+8*16*16=4096, total block block count is 32*4+8*16=256, block range is 00~FF.
	oldKey [in]	The old key, 16 bytes.
	newKey [in]	The new key, 16 bytes
Return	0: Success Other value: Error code	
Comment	None	
Example	None	

3.4.1.11 SLE4442/SLE4428

3.4.1.11.1 SLE verify password

Prototype	int sleAuthKey(byte[] key)	
Feature	SLE verify password	
Parameter	key[in]	The password,SLE4442 is 3 bytes,SLE4428 is 2 bytes,default value is all 'F'
Return	0: Success Other value: Error code	
Comment	None	

Example	None

3.4.1.11.2 SLE change password

Prototype	int sleChangeKey(in byte[] newKey)	
Feature	SLE change password	
Parameter	newKey[in]	The new password,SLE4442 is 3 bytes, SLE4428 is 2 bytes
Return	0: Success Other value: Error code	
Comment	This method only useful after calling **sleAuthKey()** sucess.	
Example	None	

3.4.1.11.3 SLE read data

Prototype	int sleReadData(int startAddress, int length, byte[] outData)	
Feature	SLE read a successive memory segment data	
Parameter	startAddress [in]	The start address
	length[in]	The data length of read
	outData[out]	Buffer, store the read data
Return	>=0: The valid data length of **outData** <0: Error code	
Comment	None	
Example	None	

3.4.1.11.4 SLE write data

Prototype	int sleWriteData(int startAddress, byte[] data)	
Feature	SLE write a successive memory segment data	
Parameter	startAddress [in]	The start address
	data[in]	The data to write, max length is 253 bytes
Return	0: Success Other value: Error code	
Comment	If card has password, this method only useful after calling **sleAuthKey()** sucess.	
Example	None	

3.4.1.11.5 SLE get remain authentication count

Prototype	int sleGetRemainAuthCount()	
Feature	SLE get remain authentication count	
Parameter	[in]	None
Return	>=0: The remain authentication count	

	<0: Error code
Comment	If the remain authentication count is 0, card are locked and can not verify password or write data.
Example	None

3.4.1.11.6 SLE write protection memory

Prototype	int sleWriteProtectionMemory(int startAddress, int length)	
Feature	SLE lock a successive memory segment, make it can not be written, no matter verify password success or not	
Parameter	startAddress[in]	The start address
	length[in]	The length of memory
Return	0: Success Other value: Error code	
Comment	None	
Example	None	

3.4.1.11.7 SLE read memory protection status

Prototype	int sleReadMemoryProtectionStatus(int startAddress, int length, byte[] dataOut)	
Feature	SLE read a successive memory protection status.	
Parameter	startAddress[in]	The start address
	length[in]	The length of memory
	dataOut[out]	Buffer, store each memory unit locking status, 0-locked, 1-not locked
Return	>=0: The valid data length of **outData** <0: Error code	
Comment	None	
Example	None	

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

3.4.1.12.1 AT24C read data

Prototype	int at24cReadData(int startAddress, int length, byte[] outData)	
Feature	AT24C read a successive memory segment data	
Parameter	startAddress[in]	The start address
	length[in]	The length of memory
	dataOut[out]	Buffer, store the read data
Return	>=0: The valid data length of **outData** <0: Error code	
Comment	None	

Example	None

3.4.1.12.2 AT24C write data

Prototype	int at24cWriteData(int startAddress, byte[] data)	
Feature	AT24C write a successive memory segment data	
Parameter	startAddress[in]	The start address
	data [in]	The data to write
Return	0: Success Other value: Error code	
Comment	None	
Example	None	

3.4.1.13 AT88SC

3.4.1.13.1 AT88SC verify password

Prototype	int at88scAuthKey(byte[] key, int rwFlag, int zoneNo)	
Feature	AT88SC verify password	
Parameter	key[in]	The password,AT88SC1608 is 3 bytes ,default value is all ‘F’
	rwFlag[in]	Write/Read flag, 0- The write key,1-The read key
	zoneNo[in]	User zone number, range 0~7
Return	0: Success Other value: Error code	
Comment	For AT88SC1608, there are 8 user zones(0~7), each zone contains 256 bytes, total 2048 bytes, address range 0~2047(0~7FF)	
Example	None	

3.4.1.13.2 AT24C88 change password

Prototype	int at88scChangeKey(byte[] newKey, int rwFlag, int zoneNo)	
Feature	AT88SC change password	
Parameter	newKey[in]	The new password,AT88SC1608 is 3 bytes
	rwFlag[in]	Write/Read flag, 0-The write key, 1-The read key
	zoneNo[in]	User zone number, range 0~7
Return	0: Success Other value: Error code	
Comment	This method only useful after calling **at88scAuthKey()** sucess.	
Example	None	

3.4.1.13.3 AT88SC read data

Prototype	int at88scReadData(int startAddress, int length, int zoneFlag, byte[] outData)	
Feature	AT88SC read a successive memory segment data	
Parameter	startAddress [in]	The start address
	length[in]	The data length of read
	zoneFlag[in]	The zone flag, 0-Config zone,1-User zone
	outData[out]	Buffer, store the read data
Return	>=0: The valid data length of **outData** <0: Error code	
Comment	None	
Example	None	

3.4.1.13.4 AT88SC write data

Prototype	int at88scWriteData(int startAddress, int zoneFlag, byte[] dataIn)	
Feature	SLE write a successive memory segment data	
Parameter	startAddress [in]	The start address
	zoneFlag[in]	The zone flag, 0-Config zone,1-User zone
	data[in]	The data to write, max length is 253 bytes
Return	0: Success Other value: Error code	
Comment	If card has password, this method only useful after calling **at88scAuthKey()** sucess.	
Example	None	

3.4.1.13.5 AT88SC get remain authentication count

Prototype	int at88scGetRemainAuthCount(int rwFlag, int zoneNo)	
Feature	AT88SC get remain authentication count	
Parameter	rwFlag[in]	Write/Read flag, 0-The write key, 1-The read key
	zoneNo[in]	User zone number, range 0~7
Return	>=0: The remain authentication count <0: Error code	
Comment	If the remain authentication count is 0, card are locked and can not verify password or write data.	
Example	None	

3.4.1.14 Transmit APDU command to card(extended method)

Prototype	int transmitApduEx(int cardType, byte[] sendBuff, out byte[] recvBuff)

Feature	Transmit the **sendBuff** to card directly. This function support the general interface protocol for IC card.(T=0 and T=1), and also support the general interface protocol for contactless card. (T=CL)	
Parameter	cardType[in]	Card type currently in operation
	sendBuff[in]	Data which want to transmit to card. Max length is 255B
	recvBuff [out]	Data received from card. The max receive data length is 255B, recvBuff.lenght>=255
Return	>=0: The valid data length in recvBuff <0: Error code	
Comment	This method is different from **transmitApdu** only when cardtype is **Mifare**,the following text show the differences: 1. When card type is **Mifare**, the first byte(B1) of send data represents the setting of communication parameters: 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 - Send whole data of the last byte, TxLastBits = n (n≠0) - Send n bits of the last byte. 2.For interface **transmitApdu()**, param **sendBuff** should **not contains** B1. SDK defaults to send **0x03 + sendBuff** to card. 3. For interface **transmitApduEx()**, param sendBuff should **contains** B1, sendBuff[0] is B1,the format of B1 should comply to paragraph 1.	

3.4.1.15 CTX512B

3.4.1.15.1 CTX512B read block data

Prototype	int ctx512ReadBlock(int block, byte[] outData)	
Feature	CTX512B card read block data	
Parameter	block[in]	Block number
	outData[out]	Buffer, store the read data (2B)
Return	>=0: The valid data length of **outData** <0: Error code	
Comment	None	

Example	None

3.4.1.15.2 CTX512B write block data

Prototype	int ctx512WriteBlock(int block, byte[] data)	
Feature	CTX512B card write block data	
Parameter	block[in]	Block number
	data [in]	The data to be written(2B)
Return	0: Success Other value: Error code	
Comment	None	
Example	None	

3.4.1.15.3 CTX512B update block data

Prototype	int ctx512UpdateBlock(int block, byte[] data)	
Feature	CTX512B card update block data	
Parameter	block[in]	Block number
	data[in]	The data to be updated(2B)
Return	0: Success Other value: Error code	
Comment	None	
Example	None	

3.4.1.15.4 CTX512B get signature data

Prototype	int ctx512GetSignature(int block, byte[] random, byte[] outData)	
Feature	CTX512B card get signature data	
Parameter	block[in]	Block number
	random[in]	Random data (6B)
	outData[out]	Buffer, store the signature data (2B)
Return	>=0: The valid data length of **outData** <0: Error code	
Comment	None	
Example	None	

3.4.1.15.5 CTX512B read 4 successive blocks data

Prototype	int ctx512MultiReadBlock(int startBlock, byte[] outData)	
Feature	CTX512B card read 4 successive blocks data	
Parameter	startBlock[in]	The start block number
	outData[out]	Buffer, store the read data (8B)

Return	>=0: The valid data length of **outData** <0: Error code
Comment	None
Example	None

3.4.1.16 Check card(extended method)

Prototype	void checkCardEx(int cardType, int ctrCode, int stopOnError, CheckCardCallbackV2 checkCardCallback, int timeout)	
Feature	For card checking, magnetic stripe card, IC card and NFC card are supported. After checking, the card type will be put into CheckCardCallbackV2.	
Parameter	cardType[in]	Composite card types, support checking multi cards. This param can be a composite value of CardType.getValue(), for example, check MAGNETIC,NFC,IC at the same time, set this param as: CardType.MAGNETIC.getValue()|CardType.NFC.getValue()| CardType.IC.getValue()
	ctrCode[in]	Card active control code Bit0-bit1: Contact card working voltage 0: VCC_3000mV 1: VCC_1800mV 2: VCC_5000mV 3: Reserved Bit2: The speed of reset on Contact CPU card or SAM card 0: SPD_1X 1: SPD_4X Bit3: support PPS or not 0: NOT SUPPORT 1: SUPPORT Bit4: The procedure of reach agreement on contact CPU card or SAM card 0: ICC_SPEC 1: ICC_EMV
	stopOnError[in]	Whether to stop immediately when checking special card type error
	callback[in]	Check card callback. Refer to CheckCardCallbackV2
	timeout[in]	Timeout (unit: second). Effective time range: 1-120s
Return	None	
Comment	No distinction between bank card and non bank card.	

3.4.1.17 Transmit APDU command to card(second time extended method)

Prototype	int transmitApduExx(int cardType, int ctrCode, byte[] sendBuff, byte[] recvBuff);
Feature	Transmit the **sendBuff** to card directly.

	This function support the general interface protocol for IC card.(T=0 and T=1), and also support the general interface protocol for contactless card. (T=CL)	
Parameter	cardType[in]	Card type currently in operation
	ctrCode[in]	Card active control code Bit0-bit1: Contact card working voltage 0: VCC_3000mV 1: VCC_1800mV 2: VCC_5000mV 3: Reserved Bit2: Reserved Bit3: The speed of reset on Contact CPU card or SAM card 0: SPD_1X 1: SPD_4X Bit4: support PPS or not 0: NOT SUPPORT 1: SUPPORT Bit5: The procedure of reach agreement on contact CPU card or SAM card 0: ICC_SPEC 1: ICC_EMV
	sendBuff[in]	Data to transmit to card. Max length is 255B
	recvBuff [out]	Data received from card. The max receive data length is 255B, recvBuff.lenght>=255
Return	>=0: The valid data length in recvBuff <0: Error code	
Comment	**If cardType is Mifare, prama sendBuff shoule has the same format as sendBuff of transmitApduEx().**	

3.4.1.18 Transmit multi APDUs to card

Prototype	int transmitMultiApdus(int cardType, int ctrCode, List<String> sendList, List<String> recvList)	
Feature	Transmit multi APDUs to card directly. At one time, the max send APDU number is 8. This function support the general interface protocol for IC card.(T=0 and T=1), and also support the general interface protocol for contactless card. (T=CL)	
Parameter	cardType[in]	Card type currently in operation
	ctrCode[in]	Card active control code Bit0-bit1: Contact card working voltage 0: VCC_3000mV 1: VCC_1800mV 2: VCC_5000mV 3: Reserved Bit2: Reserved Bit3: The speed of reset on Contact CPU card or SAM card

		0: SPD_1X 1: SPD_4X Bit4: support PPS or not 0: NOT SUPPORT 1: SUPPORT Bit5: The procedure of reach agreement on contact CPU card or SAM card 0: ICC_SPEC 1: ICC_EMV
	sendList[in]	APDU(Hex) list to transmit to card, each item is one APDU
	recvList[out]	The respond data of each sent APDU from card(Hex)
Return	>=0: The valid data length in recvBuff <0: Error code	
Comment	**If cardType is Mifare, prama sendBuff shoule has the same format as sendBuff of transmitApduEx().**	

3.4.1.19 Check card with ciphertext track data

Prototype	int checkCardEnc(Bundle bundle, CheckCardCallbackV2 checkCardCallback, int timeout)	
Feature	For card checking, magnetic stripe card, IC card and NFC card are supported. After checking, the card type will be put into CheckCardCallbackV2. **For card type CardType.MAGNETIC.getValue(), checkCardCallback return ciphertext track data**	
Parameter	bundle [in]	check card params, contains key: cardType: int, card type, composite value, eg: CardType.MAGNETIC.getValue()|CardType.NFC.getValue() |CardType.IC.getValue() encKeySystem: int, key system, refer to KEY_SYSTEM_CONSTANTS encKeyIndex: int**,** TDK (Track data key) index encMode: int, Encryption mode encIv: byte[], IV(initial vector), if **encMode** is ECB，value is **null**, otherwise value is 8 bytes data encPaddingMode: byte, The padding mode on encryption encMaskStart: int, 0~6, how much characters are plaintext in front of PAN encMaskEnd: int, 0~4, how much characters are plaintext in rear of PAN encMaskWord: char, 0 or non-digit character，The mask character for **EncMaskStart**~**encMaskWord** PAN, default is '*' ctrCode: int, card active control code(default value is 0): b0~b1:00-VCC_3000mV, 01-VCC_1800mV, 02-VCC_5000mV, 03-reserved b2:0-SPD_1X,1-SPD_4X b3: Is support PPS,0-not support,1-support

		b4: contact CPU card and SAM card protocol procedure,0-ICC_SPEC,1-ICC_EMV stopOnError: int, Whether to stop immediately when checking special card type error, 0-not stop, 1-stop
	callback[in]	Check card callback. Refer to CheckCardCallbackV2
	timeout[in]	Timeout (unit: second). Effective time range: 1-120s
Return	None	
Comment	No distinction between bank card and non bank card.	

3.4.1.20 Set smart card related params

Prototype	int smartCardIoControl(int cardType, int ctrCode, byte[] dataIn, byte[] dataOut)	
Feature	Set smart card related params	
Parameter	cardType [in]	The card type
	ctrCode [in]	The control code, has following values: 0-Set system code of Felica cad polling command (default is 0xffff),2B,Big endian 1-Set contactless apdu timeout time, unit: ms, 4B, Big endian 2-Get contactless register config(TLV format) 3-Set contactless register config(TLV format) 4-Get contactless param config(TLV format) 5-Set contactless param config(TLV format)
	dataIn[in]	The input data, refer to **ctrCode**
	dataOut[out]	Buffer, store output data.
Return	>=0: The valid data length in dataOut <0: Error code	
Comment	None	
Example	None	

3.4.2 CheckCardCallbackV2

3.4.2.1 Find magnetic card

Prototype	void findMagCard(Bundle info)	
Feature	Find magnetic stripe card.	
Parameter	info[in]	Contains the following data: cardType: int, the card type TRACK1: String, track1 data TRACK2: String, track2 data TRACK3: String, track3 data pan: String, PAN data (return by checkCardEnc())

		name: String, cardhold name (return by checkCardEnc()) expire: String, card expire date (return by checkCardEnc()) servicecode: String, card service code (return by checkCardEnc()) track1ErrorCode: int, track1 error code track2ErrorCode: int, track2 error code track3ErrorCode: int, track3 error code The track error code can be one of the following values: -1: Track no data -2: Track parity check error -3: Track LRC check error
Return	None	
Comment	None	

3.4.2.2 Find IC card

Prototype	void findICCard(String atr)	
Feature	Find IC card	
Parameter	atr[in]	The card ATR
Return	None	
Comment	None	

3.4.2.3 Find radio frequency card

Prototype	void findRFCard(String uuid)	
Feature	Find RF Card	
Parameter	uuid[in]	The card UUID
Return	None	
Comment	None	

3.4.2.4 Check card error

Prototype	void onError(int code, string message)	
Feature	Check card error callback.	
Parameter	code[in]	Error code
	message[in]	Error message
Return	None	
Comment	None	

3.4.2.5 Find IC card(extended method)

Prototype	void findICCardEx(Bundle info)	
Feature	Find IC card	
Parameter	info[in]	Contains the following data: cardType: int, the checked card type atr: String, the card ATR
Return	None	
Comment	Compare with **findICCard()**, This method provide more detail info in param [info], Both this method and **findICCard()** will be called in check card process. Client app can implement one of the two to do process.	

3.4.2.6 Find radio frequency card(extended method)

Prototype	void findRFCardEx(Bundle info)	
Feature	Find RF Card	
Parameter	info[in]	Contains the following data: cardType: int, the checked card type uuid: String, the card UUID ats: String, the card ATS cardCategory: int, the card category, 'A' or 'B' atqa: byte[], the card ATQA
Return	None	
Comment	Compare with **findRFCard()**, This method provide more detail info in param [info], Both this method and **findRFCard()** will be called in check card process. Client app can implement one of the two to do process.	

3.4.2.7 Check card error(extended method)

Prototype	void onErrorEx(Bundle info)	
Feature	Check card error callback.	
Parameter	info[in]	Contains the following data: cardType: int, the failed card type code: int, the error code message: String, the error message
Return	None	

Comment	Compare with **onError()**, This method provide more detail info in param [info], Both this method and **onError()** will be called in check card process. Client app can implement one of the two to do process.

3.5 PinPad operation module

3.5.1 PinPadOpt

3.5.1.1 Initialize PinPad

Prototype	void initPinPad(PinPadConfigV2 config, PinPadListenerV2 listener)	
Feature	Initialize PinPad: import configuration parameter and register callback listener.	
Parameter	config[in]	PinPad configuration, refer to PinPadConfigV2 If set PinPadConfigV2.pinPadType as 0, an SDK built-in PinPad will be shown
	listerner[in]	Callbacks (if PinPadType in PinPadConfigV2 uses a built-in PIN keyboard). Refer to PinPadListenerV2
	[out]	None
Return	None	
Comment	 Sunmi PinBlock format.docx For Sunmi PinBlock format, please refer to **Sunmi PinBlock format**	

3.5.1.2 Cancel input PIN

Prototype	void cancelInputPin()	
Feature	Cancel input PIN.	
Parameter	[in]	None
	[out]	None
Return	None	
Comment	None	

3.5.1.3 Set PinPad showing text

Prototype	void setPinPadText(PinPadTextConfigV2 config)	
Feature	Set PinPad showing text	
Parameter	config[in]	Showing text config, refer to PinPadTextConfigV2

	[out]	None
Return	None	
Comment	None	

3.5.1.4 Set PinPad mode

Prototype	int setPinPadMode(Bundle bundle)	
Feature	Set PinPad mode	
Parameter	bundle[in]	PinPad mode config, contain the following keys: normal: int, normal mode (0-disable, 1-enable) longPressToClear: int, long press clear key to clear while inputting PIN (0-disable, 1-enable) silent: int, keep silent while inputting PIN (0-disable, 1-enable) greenLed: int, lit green LED while inputting PIN (0-disable, 1-enable)
	[out]	None
Return	0: Success Other value: Fail	
Comment	(1) The default mode is Normal mode (Key sound+ short prcess to clear + lit green LED) (2) **Normal mode** mutex to other modes, and if set normal as 1, other value will be discarded (3) The set mode only valid for next time input PIN, and after input PIN finished, SDK auto restore mode to **Normal mode**.	

3.5.1.5 Get PinPad mode

Prototype	int getPinPadMode(Bundle bundle)	
Feature	Get current PinPad mode	
Parameter	[in]	None
	bundle[out]	PinPad mode config, contain the following keys: normal: int, normal mode (0-disable, 1-enable) longPressToClear: int, long press clear key to clear while inputting PIN (0-disable, 1-enable) silent: int, keep silent while inputting PIN (0-disable, 1-enable) greenLed: int, lit green LED while inputting PIN (0-disable, 1-enable)
Return	0: Success Other value: Fail	
Comment	None	

3.5.1.6 Initialize PinPad (extened method)

Prototype	String initPinPadEx(Bundle config, PinPadListenerV2 listener);

Feature	Initialize PinPad	
Parameter	config[in]	PinPad config, contain following keys: pinPadType: int, 0-SDK built-in PinPad, 1-Client App customized PinPad pinType: int, 0-online PIN, 1-offline PIN isOrderNumKey: int, 0-random order keyboard (default), 1-in order keybaord pan: byte[], ASCII to bytes, eg:" 123456" .getBytes("US-ASCII") pinKeyIndex: int, PIK (PIN key) index minInput: int, default 0, minimum input PIN numbers maxInput: int, default 6, maximum input PIN numbers inputStep: int, default 1, input PIN step timeout: int, unit: ms, default 60000, input PIN timeout time isSupportbypass: int, 0-not support, 1-support(default) pinblockFormat: int, default 0, refer to PINBLOCK_FORMAT algorithmType: int, 0-3DES(8B PinBlock), 1-SM4(16B PinBlock), 2-AES(16B PinBlock), algorithm of ecryption PIN keySystem: int, 0-SEC_MKSK(default), 1-SEC_DUKPT diversify: byte[], 3DES PIK operate with diversify to generate new PIK, then use new PIK to calculate PinBlock
	bundle[out]	输 PIN 回调，详见 PinPadListenerV2
Return	0: Success Other value: Fail	
Comment	None	

3.5.1.7 Set PIN anti-exhaustive protection mode

Prototype	int setAntiExhaustiveProtectionMode(int level)	
Feature	Set PIN anti-exhaustive protection mode	
Parameter	level[in]	Protection level, range 1-5, each value indicates max input PIN times in specified time range: 1-2 minutes input 4 times 2-6 minutes input 12 times 3-15 minutes input 30 times 4-30 minutes input 60 times 5-60 minutes input 120 times
Return	>=0: The time should to wait before new mode take effect, 0 means new mode take effect immediately Other value: Fail	
Comment		

3.5.1.8 Get PIN anti-exhaustive protection mode

Prototype	int getAntiExhaustiveProtectionMode()

Feature	Get PIN anti-exhaustive protection mode	
Parameter	[in]	None
Return	>=0: Current PIN anti-exhaustive protection level, range 1-5 Other value: Fail	
Comment	None	

3.5.2 PinPadListenerV2

Note: This interface is the interface passed between AIDLs and must be implemented in accordance with the Aidl interface when the callback is passed.

3.5.2.1 Press number key

Prototype	void onPinLength(int len)	
Feature	Import the PIN length, which is used for display when input PIN.	
Parameter	len[in]	Length of PIN entered
Return	None	
Comment		

3.5.2.2 Press confirm key

Prototype	void onConfirm(int type, byte[] pinBlock)	
Feature	Press confirm key, return PinBlock. (1) type = 0 (online PIN): (a) pinBlock=null: bypass mode (the user did not enter pin and press confirm key directly) (b) pinBlock !=null: the returned PinBlock, length is 8B(3DES) or 16B(SM4/AES) (2) type = 1(offline PIN): pinBlock length is always 1, and pinBlock[0] is 0, pinBlock is meaningless in this case.	
Parameter	type[in]	Pin type, 0-online PIN, 1-offline PIN.
	pinBlock [out]	Pin block data
Return	None	
Comment	**Offline PIN is verified by card directly, return PinBlock to client is not necessary, SDK just return 1 byte array to mark this situation.**	

3.5.2.3 Press cancel key

Prototype	void onCancel ()	
Feature	Cancel PIN input.	
Parameter	[in]	None

Return	None
Comment	None

3.5.2.4 Input pin error

Prototype	void onError (int errorCode)	
Feature	Error message	
Parameter	errorCode [in]	Error code, refer to Error Code Definition
Return	None	
Comment	None	

3.6 Security operation module

3.6.1 Save plaintext key

Prototype	int savePlaintextKey(int keyType, byte[] keyValue, byte[] checkValue, int keyAlgType, int keyIndex)	
Feature	Save plaintext key	
Parameter	keyType[in]	Key type. Refer to: Aidl constants Key type constant definition
	keyValue[in]	Key data
	checkValue[in]	Key check value. If the key check value is null, it does not check (when keyAlgType is SM4, the value of checkValue is 16 bytes).
	keyAlgType[in]	For specifying the type of key that is currently saved, refer to appendix: Aidl constant Key algorithm type..
	keyIndex[in]	Key index , range 0-199
Return	0: Success Other value: Fail	
Comment	1.When saving TMK, Client can specify the **encryptIndex** as a TMK (which already stored) index, this can support multi-layer TMK. 2.When isEncrypt is false, the decryption key parameter is 1 by default. 3.Client should guarantee key index mutex, that is, if stored a MAC key in index 4, then store a track key in index 4 too, the MAC key will be overwritten by the track key.	

3.6.2 Save ciphertext key

Prototype	int saveCiphertextKey(int keyType, byte[] keyValue, byte[] checkValue, int encryptIndex, int keyAlgType, int keyIndex)	
Feature	Save ciphertext key	
Parameter	keyType[in]	Key Type: Reference Appendix: Aidl Constant Key Type Definition.

		KEY_TYPE_KEK Type cannot be saved in ciphertext
	keyValue[in]	Key data
	checkValue[in]	Key check value. If the key check value is null, it does not check (when keyAlgType is SM4, the value of checkValue is 16 bytes).
	encryptIndex[in]	Used to decrypt the key ciphertext index, note that here is the index of TMK.
	keyAlgType[in]	For specifying the type of key that is currently saved, refer to appendix: Aidl constant Key algorithm type.
	keyIndex[in]	Key index, range 0-199
Return	0: Success Other value: Fail	
Comment	1.When saving TMK, Client can specify the **encryptIndex** as a TMK (which already stored) index, this can support multi-layer TMK. 2.When isEncrypt is false, the decryption key parameter is 1 by default. 3.Client should guarantee key index mutex, that is, if stored a MAC key in index 4, then store a track key in index 4 too, the MAC key will be overwritten by the track key.	

3.6.3 Calculate MAC

Prototype	int calcMac (int keyIndex, int macType , byte[] dataIn, byte[] dataOut)	
Feature	Calculate MAC	
Parameter	keyIndex[in]	Key index, range 0-199
	macType[in]	MAC encryption algorithm, refer to: Aidl constant.MAC algorithm type
	dataIn[in]	The original data
	dataOut[out]	MAC result
Return	>= 0: MAC data length. < 0: Error code.	
Comment	DataIn is the packet data to be computed. Many MAC algorithms have been implemented in the SDK. The results will be returned through dataOut, and the incoming dataOut will be fixed to 8 bytes long byte such as: byte [] dataOut = new byte [8].	

3.6.4 Encrypt data

Prototype	int dataEncrypt (int keyIndex , byte[] dataIn, int encryptionMode, byte[] iv, byte[] dataOut)	
Feature	Encrypt data	
Parameter	keyIndex[in]	Key index, range 0-199
	datain[in]	The original data
	encryptionMode[in]	Encryption mode, refer to appendix: Aidl constant encryption mode constant.
	iv[in]	The initial vector, the encrypted mode is the ECB space, and the 8 byte vector for other encryption modes.

	dataout[out]	Encryption data
Return	0: Success Other value: Fail	
Comment	None	
Example	1.If encryptionMode is DATA_MODE_OFB or DATA_MODE_CFB, dataIn length could not be multiply of 8 or 16, otherwise dataIn length should be multiply of 8 or 16 2.The dataOut length should be equal to dataIn length. eg : byte[] dataIn = new byte[16]; dataOut length should be 16	

3.6.5 Decrypt data

Prototype	int dataDecrypt(int keyIndex , byte[] dataIn, int encryptionMode, byte[] iv, byte[] dataOut)	
Feature	Decrypt data	
Parameter	keyIndex[in]	Key index, range 0-199
	dataIn[in]	Ciphertext data
	encryptionMode[in]	Encryption mode, refer to appendix: Aidl constant encryption mode constant.
	iv[in]	The initial vector, the encrypted mode is the ECB space, and the 8 byte vector for other encryption modes.
	dataOut[out]	plaintext
Return	0: Success Other value: Fail	
Comment	None	
Example	1.If encryptionMode is DATA_MODE_OFB or DATA_MODE_CFB, dataIn length could not be multiply of 8 or 16, otherwise dataIn length should be multiply of 8 or 16 2.The dataOut length should be equal to dataIn length. eg : byte[] dataIn = new byte[16]; dataOut length should be 16	

3.6.6 Inject dukpt key

Prototype	int saveKeyDukpt(int keyType, byte[] keyValue, byte[] checkValue, byte[] ksn, int encryptType, int keyIndex)	
Feature	Inject dukpt key	
	keyType[in]	Key type Contain: BDK- base dispersed key: corresponding to Aidl constant class key type constant KEY_TYPE_DUPKT_BDK. IPEK- initial PIN encryption key: corresponding Aidl constant class key type constant KEY_TYPE_DUPKT_IPEK.
Parameter	keyValue[in]	Key data
	checkValue[in]	Key check value(When injecting BDK null values)

	ksn[in]	KSN
	encryptType[in]	Key algorithm type, refer to: Aidl constant Key algorithm type
	keyIndex[in]	Key index , range 3DES: 0-9 or 1100-1199, AES:10-19 or 2100-2199
Return	0: Success <0: Fail	
Comment	None	
Example	None	

3.6.7 Calculate mac (Based on the DUKPT)

Prototype	int calcMacDukpt(int keyIndex, int macType , byte[] dataIn, byte[] dataOut)	
Feature	Dukpt key calculate mac.	
Parameter	keyIndex [in]	Key index, range 3DES: 0-9 or 1100-1199, AES:10-19 or 2100-2199
	macType [in]	Mac algorithm, refer to: Aidl constant.MAC algorithm type
	dataIn [in]	The original data
	dataOut[out]	Mac result.
Return	0: Success <0: Fail	
Comment	1.When keyIndex in 1100-1199(dukpt-extension key), the length of dataIn shoule <=1016	
Example	None	

3.6.8 Encrypt data (Based on the DUKPT)

Prototype	int dataEncryptDukpt(int keyIndex, byte[] dataIn, int encryptionMode, byte[] iv, byte[] dataOut)	
Feature	Dukpt key encrypt data	
Parameter	keyIndex [in]	Key index, range 3DES: 0-9 or 1100-1199, AES:10-19 or 2100-2199
	encryptionMode[in]	Encryption mode, refer to appendix: Aidl constant encryption mode constant.
	iv[in]	The initial vector, the encrypted mode is the ECB space, and the 8 byte vector for other encryption modes.
	dataIn [in]	The original data
	dataOut[out]	Encryption data
Return	0: Success <0: Fail	
Comment	1.If encryptionMode is DATA_MODE_OFB or DATA_MODE_CFB, dataIn length could not be multiply of 8 or 16, otherwise dataIn length should be multiply of 8 or 16 2.The dataOut length should be equal to dataIn length. eg：byte[] dataIn = new byte[16]; dataOut length should be 16 3.hen keyIndex in 1100-1199(dukpt-extension key), the length of dataIn shoule <=1024	
Example	None	

3.6.9 Decrypt data (Based on the DUKPT)

Prototype	int dataDecryptDukpt(int keyIndex, byte[] dataIn, int encryptionMode, byte[] iv, byte[] dataOut)	
Feature	Dukpt Key decrypt data	
Parameter	keyIndex[in]	Key index, range 3DES: 0-9 or 1100-1199, AES:10-19 or 2100-2199
	encryptionMode[in]	Encryption mode, refer to appendix: Aidl constant encryption mode constant.
	iv[in]	The initial vector, the encrypted mode is the ECB space, and the 8 byte vector for other encryption modes.
	dataIn[in]	Encryption data
	dataOut[in]	Decryption results
Return	0: Success <0: Fail	
Comment	1.If encryptionMode is DATA_MODE_OFB or DATA_MODE_CFB, dataIn length could not be multiply of 8 or 16, otherwise dataIn length should be multiply of 8 or 16 2.The dataOut length should be equal to dataIn length. eg：byte[] dataIn = new byte[16]; dataOut length should be 16 3When keyIndex in 1100-1199(dukpt-extension key), the length of dataIn shoule <=1024	
Example	None	

3.6.10 Dukpt's KSN increased by 1

Prototype	int dukptIncreaseKSN(int keyIndex)	
Feature	Dukpt's KSN increased by 1	
Parameter	keyIndex[in]	Dukpt key index, range 3DES: 0-9 or 1100-1199, AES:10-19 or 2100-2199
Return	0: Success <0: Fail	
Comment	**Client APP should pay attention to the return value, if return value <0(failure), Client App shoule repeat call this interface again until it return 0, this can avoid failure in next time access dukpt**	
Example	None	

3.6.11 Dukpt gets the current KSN

Prototype	int dukptCurrentKSN(int keyIndex, byte[] outKSN)	
Feature	Dukpt gets the current KSN	
Parameter	keyIndex[in]	Dukpt key index, range 3DES: 0-9 or 1100-1199, AES:10-19 or 2100-2199
	outKSN[out]	buffer, store KSN. Dukpt-3DES KSN is 10 bytes, Dukpt-AES KSN is 12 bytes
Return	0: Success	

	<0: Fail
Comment	Key for Dukpt key system
Example	None

3.6.12 Get key CheckValue

Prototype	int getKeyCheckValue(int keySystem, int keyIndex, byte[] dataOut)	
Feature	Get key CheckValue	
Parameter	keySystem[in]	Key system. Reference appendix: KEY_SYSTEM_CONSTANTS
	keyIndex[in]	Key index. For keySystem SEC_DUKPT, the index range is 3DES: 0-9 or 1100-1199, AES:10-19 or 2100-2199 , for keySystem SEC_MKSK, and the index range is 0-199.
	dataOut[out]	Fixed to 4 bytes
Return	=0 : Success <0: Fail	
Comment	The key index obtained by injecting the BDK is the CheckValue of the BDK corresponding to the IPEK.	
Example	None	

3.6.13 Get TUSN encrypted data (Only for the Chinese market)

Prototype	int getTUSNEncryptData (String dataIn, byte[] dataOut)	
Feature	Get TUSN encrypted data	
Parameter	dataIn[in]	Used to calculate the scatter value of ciphertext
	dataOut[out]	Generated ciphertext
Return	0: Success Other value: Fail	
Comment	None	
Example	DataIn encryption random factor (bank card for card number 6, sweep code category for the code after 6). DataOut is fixed into 4 byte byte[] such as: byte[] dataOut = new byte[4];	

3.6.14 Generate RSA key pair

Prototype	int generateRSAKeys(int pubKeyIndex, int pvtKeyIndex, int keysize, String pubExponent)	
Feature	Generate RSA public/private key pair.	
Parameter	pubKeyIndex [in]	Public key stored index, range 0-19
	pvtKeyIndex [in]	Private key stored index, range 0-19
	keysize [in]	The key size,512~65536,Unit: bit, such as 512,1024,etc
	pubExponent [in]	The public exponent
Return	0: Success Other value: Fail	

Comment	None
Example	None

3.6.15 Get RSA public key

Prototype	int getRSAPublicKey(int pubKeyIndex, byte[] dataOut)	
Feature	Get X509 encoded RSA public key data	
Parameter	pubKeyIndex [in]	The stored index of public key, range 0-19
	dataOut[out]	The out buffer, store public key data, ANS.1 X509 standard format encoding
Return	>=0: The valid data length of dataOut <0: Fail	
Comment	None	
Example	None	

3.6.16 Encrypt data with RSA

Prototype	int dataEncryptRSA(String transformation, int keyIndex, byte[] dataIn, byte[] dataOut)	
Feature	Encrypt data with RSA	
Parameter	transformation [in]	The transformation used in cipher process. Refer to: RSA transformation definition
	keyIndex [in]	The encrypt key index, range 0-19
	dataIn [in]	Data will be encrypted
	dataOut [out]	Out buffer, store encrypted data
Return	>=0: The valid data length of dataOut <0: Fail	
Comment	None	
Example	None	

3.6.17 Decrypt data with RSA

Prototype	int dataDecryptRSA(String transformation, int keyIndex, byte[] dataIn, byte[] dataOut)	
Feature	Decrypt data with RSA	
Parameter	transformation [in]	The transformation used in cipher process. Refer to: RSA transformation definition
	keyIndex [in]	The decrypt key index, range 0-19
	dataIn [in]	Data will be decrypted
	dataOut[out]	Out buffer, store decrypted data
Return	>=0: The valid length of dataOut <0: Fail	
Comment	None	
Example	None	

3.6.18 Remove RSA key

Prototype	int removeRSAKey (int keyIndex)	
Feature	Remove stored RSA public or private key	
Parameter	keyIndex [in]	The stored index of RSA key, range 0-19
Return	0: Success Other value: Fail	
Comment	None	
Example	None	

3.6.19 Store certificate

Prototype	int storeCertificate(int certIndex, byte[] certData)	
Feature	Store certificate	
Parameter	certIndex [in]	The stored index of certificate, range 0-19
	certData [in]	The certificate data
Return	0: Success Other value: Fail	
Comment	None	
Example	None	

3.6.20 Get certificate

Prototype	int getCertificate(int certIndex, byte[] dataOut)	
Feature	Get stored certificate	
Parameter	certIndex [in]	The stored index of certificate, 0-19
	dataOut [out]	Out buffer, store certificate data
Return	>=0: The valid data length of dataOut <0: Fail	
Comment	None	
Example	None	

3.6.21 Get initialized KSN

Prototype	int dukptGetInitKSN (byte[] outKSN)	
Feature	Get the initialized KSN	
Parameter	outKSN [out]	The initialized KSN
Return	>=0: The valid data length of outKSN <0: Fail	
Comment	None	
Example	None	

3.6.22 RSA algorithm signing data

Prototype	int signingRSA(String signAlg, int pvtKeyIndex, byte[] dataIn, byte[] dataOut)	
Feature	RSA signing data	
Parameter	signAlg [in]	The signing algorithm. Refer to: RSA signature algoritm definition
	pvtKeyIndex [in]	The RSA private key index, range 0-19
	dataIn [in]	The data to be signed
	dataOut [out]	Buffer, store the signature data.
Return	>=0: The valid data length of dataOut <0: Fail	
Comment	None	
Example	None	

3.6.23 RSA algorithm verify signature

Prototype	int verifySignatureRSA(String signAlg, byte[] pubKey, byte[] srcData, byte[] signature)	
Feature	RSA verify sinature	
Parameter	signAlg [in]	The signing algorithm. Refer to: RSA signature algoritm definition
	pubKey [in]	The RSA public key, ANS.1 X509 DER encoded
	srcData [in]	The source data before signed (raw data)
	signature [in]	The signature data.
Return	0: success <0: Fail	
Comment	None	
Example	None	

3.6.24 Inject plaintext key

Prototype	int injectPlaintextKey(String targetPkgName, int keyType, byte[] keyValue, byte[] checkValue, int keyAlgType, int keyIndex)	
Feature	Save plaintext key	
Parameter	targetPkgName[in]	The target App(App which use the injected key) package name
	keyType[in]	Key type. Refer to: Aidl constants Key type constant definition
	keyValue[in]	Key data
	checkValue[in]	Key check value. If the key check value is null, it does not check (when keyAlgType is KEY_ALG_TYPE_SM4, the value of checkValue is 16 bytes).
	keyAlgType[in]	For specifying the type of key that is currently saved, refer to appendix: Aidl constant Key algorithm type.
	keyIndex[in]	Key index, range 0-199
Return	0: Success Other value: Fail	

Comment	1. Client App should make sure keyIndex is unique for different key 2. The inject key App must has the same signature with the target App(App which use the injected key), or the injected key will not be accessible for target App.

3.6.25 Inject ciphertext key

Prototype	int injectCiphertextKey(String targetPkgName,int keyType,byte[] keyValue,byte[] checkValue, int encryptIndex, int keyAlgType, int keyIndex)	
Feature	Save ciphertext key	
Parameter	targetPkgName[in]	The target App(App which use the injected key) package name
	keyType[in]	Key Type: Reference Appendix: Aidl Constant Key Type Definition. KEY_TYPE_KEK Type cannot be saved in ciphertext
	keyValue[in]	Key data
	checkValue[in]	Key check value. If the key check value is null, it does not check (when keyAlgType is KEY_ALG_TYPE_SM4, the value of checkValue is 16 bytes).
	encryptIndex[in]	Used to decrypt the key ciphertext index, note that here is the index of TMK.
	keyAlgType[in]	For specifying the type of key that is currently saved, refer to appendix: Aidl constant Key algorithm type.
	keyIndex[in]	Key index, range 0-199
Return	0: Success Other value: Fail	
Comment	1. Client App should make sure keyIndex is unique for different key 2. The inject key App must has the same signature with the target App(App which use the injected key), or the injected key will not be accessible for target App.	

3.6.26 Encrypt data (Based on the DUKPT, extended method)

Prototype	int dataEncryptDukptEx(int keySelect, int keyIndex, byte[] dataIn, int encryptionMode, byte[] iv, byte[] dataOut)	
Feature	Dukpt key encrypt data	
Parameter	keySelect[in]	The Dukpt key select, refer to Aidl constant dukpt key select
	keyIndex[in]	Key index, range 3DES: 0-9 or 1100-1199, AES:10-19 or 2100-2199
	encryptionMode[in]	Encryption mode, refer to appendix: Aidl constant encryption mode constant.
	iv[in]	The initial vector, the encrypted mode is the ECB space, and the 8 bytes vector for other encryption modes.
	dataIn [in]	The original data
	dataOut[out]	Encryption data
Return	0: Success <0: Fail	

Comment	1.If encryptionMode is DATA_MODE_OFB or DATA_MODE_CFB, dataIn length could not be multiply of 8 or 16, otherwise dataIn length should be multiply of 8 or 16 2.The dataOut length should be equal to dataIn length. eg : byte[] dataIn = new byte[16]; dataOut length should be 16 3.When keyIndex in 1100-1199(dukpt-extension key), the length of dataIn shoule <=1024
Example	None

3.6.27 Decrypt data (Based on the DUKPT, extended method)

Prototype	int dataDecryptDukptEx(int keySelect, int keyIndex, byte[] dataIn, int encryptionMode, byte[] iv, byte[] dataOut)	
Feature	Dukpt Key decrypt data	
Parameter	keySelect[in]	The Dukpt key select, refer to Aidl constant dukpt key select
	keyIndex[in]	Key index, range 3DES: 0-9 or 1100-1199, AES:10-19 or 2100-2199
	encryptionMode[in]	Encryption mode, refer to appendix: Aidl constant encryption mode constant.
	iv[in]	The initial vector, the encrypted mode is the ECB space, and the 8 byte vector for other encryption modes.
	dataIn[in]	Encryption data
	dataOut[in]	Decryption results
Return	0: Success <0: Fail	
Comment	1.If encryptionMode is DATA_MODE_OFB or DATA_MODE_CFB, dataIn length could not be multiply of 8 or 16, otherwise dataIn length should be multiply of 8 or 16 2.The dataOut length should be equal to dataIn length. eg : byte[] dataIn = new byte[16]; dataOut length should be 16 3.When keyIndex in 1100-1199(dukpt-extension key), the length of dataIn shoule <=1024	
Example	None	

3.6.28 Calculate mac (Based on the DUKPT, extended method)

Prototype	int calcMacDukptEx(int keySelect, int keyIndex, int macType, byte[] dataIn, byte[] dataOut)	
Feature	Dukpt key calculate mac.	
Parameter	keySelect[in]	The Dukpt key select, refer to Aidl constant dukpt key select
	keyIndex [in]	Key index, range 3DES: 0-9 or 1100-1199, AES:10-19 or 2100-2199
	macType [in]	Mac algorithm, refer to: Aidl constant.MAC algorithm type
	dataIn [in]	The original data
	dataOut[out]	Mac result.
Return	0: Success <0 Fail	
Comment	1.When keyIndex in 1100-1199(dukpt-extension key), the length of dataIn shoule <=1016	

Example	None

3.6.29 Verify mac (Based on the DUKPT, extended method)

Prototype	int verifyMacDukptEx(int keySelect, int keyIndex, int macType, byte[] dataIn, byte[] macData)	
Feature	Dukpt key verify mac	
Parameter	keySelect[in]	The Dukpt key select, refer to Aidl constant dukpt key select
	keyIndex [in]	Key index, range 3DES: 0-9 or 1100-1199, AES:10-19 or 2100-2199
	macType [in]	Mac algorithm, refer to: Aidl constant.MAC algorithm type
	dataIn [in]	The original data
	macData [out]	Mac data
Return	0: Success <0 Fail	
Comment	1.When keyIndex in 1100-1199(dukpt-extension key), the length of dataIn shoule <=1016	
Example	None	

3.6.30 Save TR31 key

Prototype	int saveTR31Key(byte[] keyValue, int kbpkIndex, int keyIndex)	
Feature	Save a TR31 key	
Parameter	keyValue[in]	Key data
	kbpkIndex[in]	KBPK index, range 0-199
	keyIndex[in]	Key index, range 0-199
Return	0: Success <0: Fail	
Comment	Before save any TR31 key, KBPK should be saved firstly.	
Example	None	

3.6.31 Save ciphertext key with decryption key is RSA private key

Prototype	int saveCiphertextKeyRSA(int keyType, byte[] keyValue, byte[] checkValue, int keyAlgType, int keyIndex, int encryptIndexRSA, String transformation)	
Feature	Save ciphertext key with the decryption key is RSA private key	
Parameter	keyType[in]	Key Type: Reference Appendix: Aidl Constant key type Definition. KEY_TYPE_KEK Type cannot be saved in ciphertext
	keyValue[in]	Key data
	checkValue[in]	Key check value. If the key check value is null, it does not check (when keyAlgType is KEY_ALG_TYPE_SM4, the value of checkValue is 16 bytes).
	keyAlgType[in]	For specifying the type of key that is currently saved, refer to appendix: Aidl constant key algorithm type.

	keyIndex[in]	Key index, range 0-199
	encryptIndexRSA[in]	RSA private key index, range 0-19
	Transformation[in]	RSA transformation, refer to appendix: RSA transformation definition
Return	0: Success Other value: Fail	
Comment	None	

3.6.32 Save RSA key

Prototype	int saveRSAKey(int keyType, in byte[] keyValue, int keyIndex)	
Feature	Save RSA key	
Parameter	keyType[in]	Key Type, 0-RSA public key, 1-RSA private key
	keyValue[in]	Key data, keyType=0- ANS.1 X509 standard format encoding, keyType=1-ANS.1 PKCS#8 standard format encoding
	keyIndex[in]	Key index, range 0-19
Return	0: Success <0: Fail	
Comment	None	
Example	None	

3.6.33 Delete key

Prototype	int deleteKey(int keySystem, int keyIndex)	
Feature	Delete a stored key	
Parameter	keySystem[in]	Key system. Reference appendix: Aidl constant class key system constants
	keyIndex[in]	Key index. When keySystem is SEC_DUKPT, the index range is 3DES: 0-9 or 1100-1199, AES:10-19 or 2100-2199 when keySystem is SEC_MKSK the index range is 0-199 when keySystem is SEC_RSA_KEY, the index range is 0-19.
Return	0: Success <0: Fail	
Comment	None	
Example	None	

3.6.34 Save dukpt-AES key

Prototype	int saveKeyDukptAES(int dukptKeyType, int keyType, byte[] keyValue, byte[] checkValue, in byte[] ksn, int encryptType, int keyIndex)	
Feature	Inject dukpt key	
Parameter	dukptKeyType[in]	Dukpt key type, refer to Aidl constants DukptKeyType
	keyType[in]	Key type Contain: BDK- base dispersed key: corresponding to Aidl constant class key

		type constant KEY_TYPE_DUPKT_BDK. IPEK- initial PIN encryption key: corresponding Aidl constant class key type constant KEY_TYPE_DUPKT_IPEK.
	keyValue[in]	Key data
	checkValue[in]	Key check value(defalt 8 bytes 0)
	ksn[in]	KSN(12 bytes, first 8 bytes is valid)
	encryptType[in]	Key algorithm type, refer to: Aidl constant Key algorithm type
	keyIndex[in]	Key index ,range 10-19 or 2100-2199
Return	0: Success <0: Fail	
Comment	None	
Example	None	

3.6.35 Calculate MAC(extended method)

Prototype	int calcMacEx (int keyIndex, int keyLen, int macAlgType, byte[] diversify, byte[] dataIn, byte[] dataOut)	
Feature	Calculate MAC	
Parameter	keyIndex[in]	Mac key index ,range 0-199
	keyLen[in]	The key length, 3DES key allow use first 8/16/24 bytes to calculate Mac 0-the whole key, N- first N(8/16/24) bytes
	macAlgType[in]	Mac encryption algorithm, refer to: Aidl constant.MAC algorithm type
	Diversify[in]	Diversify factor. Before calculate Mac, Mac key operate with diversify factor to generate new Mac key, and use the new key to calculate Mac. Length of diversify is 8B/16B
	dataIn[in]	The original data
	dataOut[out]	Mac result
Return	>= 0: MAC data length. < 0: Error code.	
Comment	Mac length depends on Mac key type, 3DES key return 8B Mac, SM4 return 16B Mac.	

3.6.36 Generate SM2 key pair

Prototype	int generateSM2Keypair(int pvkIndex, Bundle pubKey)	
Feature	Generate SM2 public/private key pair	
Parameter	pvkIndex[in]	The private key index, range 0-9
	pubKey[out]	Contain following keys : data: byte[] (64B), public key data kcv: byte[] (5B), public key kcv rfu: byte[] (10B), RFU data
Return	0: Success <0: Fail	

Comment	None
Example	None

3.6.37 Inject SM2 key

Prototype	int injectSM2Key(int keyIndex, Bundle keyData)	
Feature	Inject a SM2 to HSM	
Parameter	keyIndex[in]	Key index, range 0-9
	keyData[in]	Contain following keys： data: byte[] (public key 64B/private key 32B), public/private key data kcv: byte[] (5B), public/private key kcv rfu: byte[] (10B), public/private key RFU data
Return	0: Success <0: Fail	
Comment	None	
Example	None	

3.6.38 SM2 signing data

Prototype	int sm2Sign(int pukIndex, int pvkIndex, byte[] userId, byte[] dataIn, byte[] dataOut)	
Feature	SM2 algorithm sign data	
Parameter	pukIndex[in]	The public key index, range 0-9
	pvkIndex[in]	The private key index, range 0-9
	userId[in]	The signer Id, less than 512B, recommend value is 0x31,0x32,0x33,0x34,0x35,0x36,0x37,0x38,0x31,0x32,0x33,0x34,0x35,0x36,0x37,0x38
	dataIn[in]	The data to be signed
	dataOut[out]	Buffer, store the signature data.
Return	>=0: The valid data length of dataOut <0: Fail	
Comment	None	
Example	None	

3.6.39 SM2 verify signature

Prototype	int sm2VerifySign(int pukIndex, byte[] userId, byte[] srcData, byte[] signData)	
Feature	SM2 algorithm verify signature	
Parameter	pukIndex[in]	The public key index, range 0-9
	userId[in]	The signer Id, less than 512B
	srcData[in]	The source data before signed (raw data)
	signData[in]	The signature data
Return	0: Success	

	<0: Fail
Comment	None
Example	None

3.6.40 SM2 encrypt data

Prototype	int sm2EncryptData(int pukIndex, byte[] dataIn, byte[] dataOut)	
Feature	SM2 encrypt data	
Parameter	pukIndex[in]	The public key index, range 0-9
	dataIn[in]	The data to be encrypted
	dataOut[out]	Buffer, store encrypted data
Return	>=0: The valid data length of dataOut <0: Fail	
Comment	None	
Example	None	

3.6.41 SM2 decrypt data

Prototype	int sm2DecryptData(int pvkIndex, byte[] dataIn, byte[] dataOut)	
Feature	SM2 decrypt data	
	pvkIndex[in]	The private key index, range 0-9
Parameter	dataIn[in]	The data to be decrypted
	dataOut[out]	Buffer, store decrypted data
Return	>=0: The valid data length of dataOut <0: Fail	
Comment	None	
Example	None	

3.6.42 Calculate hash

Prototype	int calcSecHash(int mode, byte[] dataIn, byte[] dataOut)	
Feature	Calculate hash	
Parameter	mode [in]	The hash type, refer to HASH_TYPE
	dataIn[in]	The data to be hashed, less than 1920B
	dataOut[out]	Buffer, store hash data
Return	>=0: The valid data length of dataOut <0: Fail	
Comment	None	
Example	None	

3.6.43 Verify Mac

Prototype	int verifyMac(int keyIndex, int macAlgType, byte[] dataIn, byte[] mac)	
Feature	Calculate hash	
Parameter	keyIndex [in]	The mac key index, range 0-199
	macAlgType [in]	Mac algorithm, refer to: Aidl constant.MAC algorithm type
	dataIn [in]	Raw input data
	mac[in]	Mac data
Return	0: Success <0: Fail	
Comment	None	
Example	None	

3.6.44 Generate RSA key pair(Only support 1024/2048 bit key)

Prototype	int generateRSAKeypair(int pvkIndex, int keySize, String pubExponent, byte[] dataOut)	
Feature	Generate RSA key pair(only support 1024/2048 bit key)	
Parameter	pvkIndex[in]	Private key index, range 0-19
	keySize[in]	Key size, 1024 or 2048
	pubExponent [in]	Public key exponent(HEX,03 or 010001)
	dataOut [out]	Buffer, store output public key module
Return	>=0: The valid data length of dataOut < 0: Error code.	
Comment	None	
Example	None	

3.6.45 Inject RSA key(Only support 1024/2048 bit key)

Prototype	int injectRSAKey(int keyIndex, int keySize, String module, String exponent)	
Feature	Inject RSA key(only support 1024/2048 bit key)	
Parameter	keyIndex[in]	Public/Private key index, range 0-19
	keySize[in]	Key size, 1024 or 2048
	module [in]	Module (Hex)
	exponent [in]	Exponent (Hex)
Return	0: Success < 0: Error code.	
Comment	None	
Example	None	

3.6.46 RSA public key encrypt/private key decrypt

Prototype	int rsaEncryptOrDecryptData(int keyIndex, int padding, byte[] dataIn, byte[] dataOut)	
Feature	RSA public key encrypt/private key decrypt	
Parameter	keyIndex[in]	Public/Private key index, range 0-19
	padding[in]	Padding mode, 0-NoPadding, 1-PKCS1Padding, 2-PKCS7Padding
	dataIn[in]	Input data, less than 896B
	dataOut[in]	Buffer, store output data
Return	>=0: The valid data length of dataOut < 0: Error code.	
Comment	None	
Example	None	

3.6.47 Get key CheckValue(extended method)

Prototype	int getKeyCheckValueEx(Bundle bundle, byte[] dataOut)	
Feature	Get key CheckValue	
Parameter	bundle[in]	Contains following keys: keySystem: int, key system keyIndex: int, key index kcvMode: int, kcv mode targetAppPkgName: String, target app package name
	dataOut[in]	Buffer, store kcv(4B)
Return	0: Success < 0: Error code.	
Comment	None	
Example	None	

3.6.48 Delete key(extended method)

Prototype	int deleteKeyEx(Bundle bundle)	
Feature	Delete key	
Parameter	bundle[in]	Contains following keys: keySystem: int, key system keyIndex: int, key index targetAppPkgName: String, target app package name
	dataOut[in]	Buffer, store kcv(4B)
Return	0: Success < 0: Error code.	
Comment	None	
Example	None	

3.6.49 Inject ciphertext key(extended method)

Prototype	int injectCiphertextKeyEx(Bundle bundle)	
Feature	Delete key	
Parameter	bundle[in]	Contains following keys: targetAppPkgName: String, target app package name keyType: int, key type(eg: KEK/TMK/PIK/TDK/MAK/REC) keyValue: byte[], key value checkValue: byte[], key check value encryptIndex: int, the decrypt key index keyAlgType: int, key algorithm type，1-3Des，2-AES，3-SM4 keyIndex: int, key index, range 0-199 keyLength：int, key length
	dataOut[in]	Buffer, store kcv(4B)
Return	0: Success < 0: Error code.	
Comment	None	
Example	None	

.6.50 Inject dukpt key(extended method)

Prototype	int injectKeyDukptEx(in Bundle bundle)	
Feature	Inject dukpt key	
Parameter	bundle[in]	Contains following keys: targetAppPkgName: String, target app package name keyValue: byte[], key value checkValue: byte[], key check value ksn：byte[], KSN encryptIndex: int, the decrypt key index keyAlgType: int, key algorithm type，1-3Des，2-AES，3-SM4 keyIndex: int, key index, 3DES: 0-9 or 1100-1199, AES: 10-19 or 2100-2199 isEncrypt：bool, is ciphertext key keyLength：int, key length
Return	0: Success < 0: Error code.	
Comment	None	
Example	None	

3.7 EMV operation module

3.7.1 EMV methods

3.7.1.1 Add AID parameter

Prototype	int addAid(AidV2 aid)	
Feature	Add or update one AID	
Parameter	aid [in]	Refer to AidV2
Return	0: Success Other value: Fail	
Comment		

3.7.1.2 Delete AID parameter

Prototype	int deleteAid(String tag9F06Value)	
Feature	Delete one AID according to the value of tag 9F06 , if you want to delete all AIDs , set tag9F06Value as null.	
Parameter	tag9F06Value [in]	The value of tag 9F06 (hex format).
Return	0: Success Other value: Fail	
Comment		

3.7.1.3 Add CAPK parameter

Prototype	int addCapk(CapkV2 capk)	
Feature	Add or update one CAPK	
Parameter	capk [in]	Refer to CapkV2
Return	0: Success Other value: Fail	
Comment		

3.7.1.4 Delete CAPK parameter

Prototype	int deleteCapk(String tag9F06Value, String tag9F22Value)	
Feature	Delete one CAPK according to the value of tag 9F06 and value of tag 9F22, if you want to delete all CAPKs, set tag9F06Value as null.	
Parameter	tag9F06Value [in]	The value of tag 9F06 (hex format)
	tag9F22Value [in]	The value of tag 9F22 (hex format)

Return	0: Success Other value: Fail
Comment	

3.7.1.5 Set terminal parameter

Prototype	int setTerminalParam(EmvTermParamV2 termParam)	
Feature	Set terminal parameter.	
Parameter	termParam [in]	Reference to EMVTermParamV2
Return	0: Success Other value: Fail	
Comment		

3.7.1.6 Check whether AID and CAPK exist or not

Prototype	int isExistCapkAndAid()	
Feature	Determine the existence of CAPK and AID.	
Parameter	[in]	None
Return	-1: Both AID and CAPK not exist. 0: Both AID and CAPK exist. 1: Only exist AID 2: Only exist CAPK	
Comment	None	

3.7.1.7 Initialize EMV process

Prototype	int initEmvProcess ()	
Feature	Initialize EMV process	
Parameter	[in]	None
Return	0: Success Other value: Fail	
Comment	None	

3.7.1.8 Strat transaction process

Prototype	void transactProcess(EMVTransDataV2 transData, EMVListenerV2 listener)	
Feature	Start EMV process	
Parameter	transData [in]	Refer to EMVTransDataV2
	listener [in]	Refer to EMVListenerV2
Return	None	

Comment	

3.7.1.9 Read one TLV data from kernel

Prototype	int getTlv(int opCode,String tag, byte[] outData)	
Feature	Read one TLV data from kernel	
Parameter	opCode[in]	TLV operation code, refer to TLV_OPERATITON_TYPE
	tag [in]	The tag to read (hex format), eg:" 95"
	outData[out]	Buffer, store the read data.
Return	>=0: The valid data length of outData Other value: Fail	
Comment		

3.7.1.10 Read TLV data list from kernel

Prototype	int getTlvList(int opCode,String[] tags, byte[] outData)	
Feature	Read TLV data list from kernel	
Parameter	opCode[in]	TLV operation code, refer to TLV_OPERATITON_TYPE
	tags [in]	The tag list to read (hex format), eg:{ "95" ," 9F2A" }
	outData [out]	Buffer , store the read data.
Return	>=0: The valid data length of outData Other value: Fail	
Comment	If there is no data has been read for a special tag, outData do not contains this tag, eg: tags={ "95" ," 9F2A" }, if no data has been read for tag "95" , outData may like as "9F2A03A00001" ,which don not contains tag "95" .	

3.7.1.11 Set TLV data to kernel

Prototype	void setTlv(int opCode,String tag, String hexValue)	
Feature	Set TLV data to kernel	
Parameter	opCode[in]	TLV operation code, refer to TLV_OPERATITON_TYPE
	tag [in]	The tag to set (hex format)
	hexValue [in]	The value corresponding to this tag (hex format)
Return	None	
Comment	None	

3.7.1.12 Set TLV data list to kernel

Prototype	void setTlvList(int opCode,String[] tags, String[] hexValues)	
Feature	Set TLV data to kernel	
Parameter	opCode[in]	TLV operation code, refer to TLV_OPERATITON_TYPE

	tags [in]	The tag list to set (hex format), eg:{ “9F1A” ,” 9F33” }
	hexValues [in]	The values corresponding to this tags list (hex format) eg:{ “0156” , “E0F8C8” }
Return	None	
Comment	None	

3.7.1.13 Import App select result to EMV

Prototype	void importAppSelect (int selectIndex)	
Feature	Import app select result to EMV procedure	
Parameter	selectIndex [in]	The selected app index, start from 0.
Return	None	
Comment	None	

3.7.1.14 Import app final select result to EMV

Prototype	void importAppFinalSelectStatus(int status)	
Feature	Import app final select result to EMV procedure.	
Parameter	status [in]	Final select result, 0:Success, 1:Fail
Return	None	
Comment	None	

3.7.1.15 Import card confirm result to EMV

Prototype	void importCardNoStatus(int status)	
Feature	Import card confirm result to EMV procedure.	
Parameter	status [in]	Confirm result, 0:Success, 1:Fail
Return	None	
Comment	None	

3.7.1.16 Import certificate authorize result to EMV

Prototype	void importCertStatus(int status)	
Feature	Import certificate authorize result to EMV procedure	
Parameter	status [in]	Authorize result, 0: Success, 1: Fail
Return	None	
Comment	None	

3.7.1.17 Import PIN input result to EMV

Prototype	void importPinInputStatus(int pinType, int inputResult)	
Feature	Import PIN input result to EMV procedure	
Parameter	pinType [in]	PIN type, 0: online PIN, 1: offline PIN
	inputResult [in]	PIN input results. 0: Success, 1: PIN cancels, 2: PIN skipped, 3: PINPAD fails.
Return	None	
Comment	None	

3.7.1.18 Import online process result to EMV

Prototype	int importOnlineProcStatus(int status, String[] tags, String[] hexValues, byte[] outData)	
Feature	Import online process result to EMV procedure.	
Parameter	status [in]	Online result, 0-Online approval, 1-Online denial, 2- Online failed
	tags[in]	Online data which need to import to kernel,eg: { "71", "72", "91", "8A", "89" }
	hexValues[in]	Values corresponding to each tag in parameter **tags**
	outData[out]	Result data of kernel process.
Return	>=0: The valid data length in **outData** <0: Error code	
Comment	None	

3.7.1.19 Import signature result to EMV

Prototype	void importSignatureStatus(int status)	
Feature	Import signature result to EMV procedure.	
Parameter	status [in]	process result, 0:Success, 1:Fail
Return	None	
Comment	None	

3.7.1.20 Read transaction log

Prototype	int readTransLog(int logType, List<String> infoOut)	
Feature	Read card transaction log	
Parameter	logType [in]	log type,0:transaction log, 1: trap log
	infoOut [out]	Buffer, store the read logs.
Return	0: Success Other value: Fail	

Comment	

3.7.1.21 Abort transact process

Prototype	void abortTransactProcess()	
Feature	Abort the emv transact process	
Parameter	[in]	None
Return	None	
Comment	**Note: If EMV process not started, call this method has no effect, if EMV process already started, call this method can interrupt the process and result in EMVListenerV2.onTransResult() called**	

3.7.1.22 Import data exchange status to emv

Prototype	void importDataExchangeStatus(int status)	
Feature	Import data exchange status to emv procedure.	
Parameter	status[in]	Data exchange result, 0:Success, 1:Fail
Return	None	
Comment	None	

3.7.1.23 Start transaction process(extended method)

Prototype	void transactProcessEx(Bundle transData, EMVListenerV2 listener)	
Feature	Start EMV process, refer to **transactProcess(EMVTransDataV2 transData, EMVListenerV2 listener)**	

Parameter	transData [in]	Emv transaction data, similar to EMVTransDataV2, this param can be set with following code:
		Bundle bundle = new Bundle();
		bundle.putString("amount", amount); //transaction amount
		bundle.putString("transType", transType);//transaction type
		bundle.putInt("flowType", flowType);//flow type
		bundle.putInt("cardType", cardType);//card type
		bundle.putString("cashbackAmount",cashbackAmount);//cashback amount
		bundle.putInt("emvAuthLevel",level);//auth level,0-Normal,1-EC PBOC

	listener[in]	Refer to EMVListenerV2
Return	None	
Comment	None	

3.7.1.24 Query electronic cash balance

Prototype	int queryECBalance(Bundle bundle)	
Feature	Query electronic cash balance	
Parameter	bundle[out]	Contains the following data: 9F51: String, Application currency code (Hex format) 9F79: long, Electronic cash balance
Return	None	
Comment	None	

3.7.1.25 Add DRL LimitSet

Prototype	int addDrlLimitSet(DrlV2 drl)	
Feature	Query electronic cash balance	
Parameter	drl[in]	DRL LimitSet, refer to DrlV2
Return	0: Success Other value: Fail	
Comment	None	

3.7.1.26 Delete DRL LimitSet

Prototype	int deleteDrlLimitSet(String programId)	
Feature	Delete one DRL LimitSet by programId, if want to delete all DRL LimitSets, set programId as null	
Parameter	programId [in]	The program ID
Return	0: Success Other value: Fail	
Comment	None	

3.7.1.27 Set terminal param(extended method)

Prototype	void setTermParamEx(Bundle bundle)	
Feature	Set terminal param	
Parameter	bundle[out]	Contains the following data: supportDRL: boolean, Is support DRl function downloadAidParam: boolean, Is support download AidParam downloadAidParamAll：boolean，Is download AidParamAll downloadPreParamEP：boolean，Is download PreParamEP optOnlineRes：boolean，Is optimize Online Result ledLightingDuration：int，Led Lighting Duration contactlessManualSelApp：boolean, Contactless transaction select App

		manually contactlessManualSelAppGeneral :boolean, Contactless transaction select App manually(Generally version) supportEP: boolean，Is supportEP importScriptData：boolean，Online denial import script data
Return	None	
Comment	None	

3.7.1.28 Query All Aid or Capk list

Prototype	int queryAidCapkList(int type, List<String> list)	
Feature	Query all Aid or Capk list in SDK	
Parameter	type[in]	The query type, 0-query all Aids 1-query all capks
	list[out]	The queried Aid or Capk list
Return	0: Success Other value: Fail	
Comment	None	

3.7.1.29 EMV transact pre process

Prototype	int transactPreProcess()	
Feature	EMV pre process	
Parameter	[in]	None
	[out]	None
Return	0: Success Other value: Fail	
Comment	None	

3.7.1.30 Add RevocationList

Prototype	int addRevocList(RevocListV2 revocList)	
Feature	Add or update a RevocationList	
Parameter	revocList[in]	A revocationList, refer to Aidl constant. RevocListV2
	[out]	None
Return	0: Success Other value: Fail	
Comment	None	

3.7.1.31 delete RevocationList

Prototype	int deleteRevocList(RevocListV2 revocList)

Feature	Delete one or all RevocationLists	
Parameter	revocList[in]	The RevocationList to be delete, refer to Aidl constant. RevocListV2. If set this param as null, all RevocationLists will be deleted
	[out]	None
Return	0: Success Other value: Fail	
Comment	None	

3.7.1.32 Set system time

Prototype	int sysSetTime(long timeStamp)	
Feature	Set system time	
Parameter	timeStamp[in]	The timestamp to be set, unit: ms
	[out]	None
Return	0: Success Other value: Fail	
Comment	None	

3.7.1.33 Get system time

Prototype	int sysGetTime(byte[] outData)	
Feature	Get system time, unit: s	
Parameter	[in]	None
	outData[out]	Buffer, store the got system time, this value is a yyyyMMddHHmmss string converted byte array, eg: hexStringToBytes(“20191130142020”)
Return	>=0: The valid data length in **outData** <0: Fail	
Comment	None	

3.7.1.34 Clear data

Prototype	int clearData(int opCode)	
Feature	Clear emv data	
Parameter	opCode[in]	Operation code, 0-clear all data, 1-clear terminal data, 2-clear card data. Refer to Aidl contact. EMV clear data constant definition
	[out]	None
Return	0: Success Other value: Fail	
Comment	None	

3.7.1.35 Set account data security param

Prototype	int setAccountDataSecParam(Bundle bundle)	
Feature	Set account data secure param	
Parameter	bundle[in]	encKeySystem: int, key system, refer to KEY_SYSTEM_CONSTANTS encKeyIndex: int, track data key index, general incoming TDK/ DUKPT index encMode: int, Encryption mode encIv: byte[], IV(initial vector), if **encMode** is ECB，value is **null**, otherwise value is 8 bytes data encPaddingMode: byte, The padding mode on encryption. If TDK is 3DES key, and PAN length is not multiple of 8, then fill PAN to the multiple of 8 by EncPaddingMode encMaskStart: int, 0~6, how much characters are clear text in front of PAN encMaskEnd: int, 0~4, how much characters are clear text in rear of PAN encMaskWord: char, 0 or non-digit character , The mask character for **EncMaskStart**~**encMaskWord** PAN, default is '*' sred: boolean, the default is true
	[out]	None
Return	0: Success Other value: Fail	
Comment	None	

3.7.1.36 Get account security data

Prototype	int getAccountSecData(int opCode, String[] tags, Bundle bundle)	
Feature	Get account security data	
Parameter	opCode[in]	TLV operation code, refer to TLV_OPERATITON_TYPE
	tags[in]	The tag list to read (hex format), eg: { "95" ," 9F2A" }
	bundle[out]	Contains the following key: Tag+Enc: String, eg: { "5AEnc, "57Enc" }, tag encryption data panMask: String, PAN masking data, only exist for tag 5A/57
Return	0: Success Other value: Fail	
Comment	None	

3.7.1.37 Import terminal risk management result

Prototype	void importTermRiskManagementStatus(int status)

Feature	Import terminal risk management result	
Parameter	status[in]	Terminal risk management result, 0-success, other value-failed
Return	None	
Comment	None	

3.7.2 EMVListenerV2 callback methods

Note: This interface is the interface passed between AIDLs and must be implemented in accordance with the Aidl interface when the callback is passed.

3.7.2.1 Request App select

Prototype	void onWaitAppSelect(List<EMVCandidateV2> candList, boolean isFirstSelect)	
Feature	If card has more than one app, request to select one app. Refer to EMVCandidateV2	
Parameter	candList [in]	The app list for selecting
	isFirstSelect[in]	Is first time to select
Return	None	
Comment		

3.7.2.2 Request app final select

Prototype	void onAppFinalSelect (String tag9F06Value)	
Feature	Notify client the emv app final select	
Parameter	tag9F06Value [in]	The value of tag 9F06
Return	None	
Comment		

3.7.2.3 Request Confirm Card number

Prototype	void onConfirmCardNo(String cardNO)	
Feature	Request confirm card number.	
Parameter	cardNO[in]	The card number to be confirmed
Return	None	
Comment		

3.7.2.4 Request confirm certificate info

Prototype	void onCertVerfiy(int certType, String certInfo)	
Feature	Request confirm certificate info if EMV process required	
Parameter	certType[in]	Certificate type

	certInfo[in]	Certificate info
Return	None	
Comment	None	

3.7.2.5 Request input PIN

Prototype	void onRequestShowPinPad(int pinType, int remainTimes)	
Feature	Request input PIN	
Parameter	pinType[in]	PIN type, 0:online PIN，1: offline PIN
	remainTimes[in]	The remain try times of offline PIN. If current is online PIN, this value is always -1, if it is the first time to try input PIN, this value is -1 too.
Return	None	
Comment	None	

3.7.2.6 Requset signature

Prototype	void onRequestSignature()	
Feature	Request user signature	
Parameter	[in]	None
Return	None	
Comment	None	

3.7.2.7 Requset online process

Prototype	void onOnlineProc()	
Feature	Request online process	
Parameter	[in]	None
Return	None	
Comment	None	

3.7.2.8 Card data exchange complete

Prototype	void onCardDataExchangeComplete()	
Feature	The kernel and card data exchange complete	
Parameter	[in]	None
Return	None	
Comment	None	

3.7.2.9 EMV procedure result

Prototype	void onTransResult(int code, String desc)	
Feature	Send EMV procedure result to client. This method is mutex with onConfirmationCodeVerified(), that is, in an EMV procedure, only one of the 2 methods will be called.	
Parameter	code [in]	The result code: 0-Success, 1-Offline approval, 2-Offline denial, <0-Error code.
	desc[in]	The result message corresponding to result code.
Return	None	
Comment	None	

3.7.2.10 Confirmation code verified

Prototype	void onConfirmationCodeVerified()	
Feature	The confirmation code has been verified. This method is mutex with onTransResult(), that is, in an EMV procedure, only one of the 2 methods will be called. **At this moment, only PayPass Contactless may call this method.**	
Parameter	[in]	None
Return	None	
Comment	None	

3.7.2.11 Request data exchage

Prototype	void onRequestDataExchange(String cardNo)	
Feature	Request user exchange data with emv procedure(MIR use this method)	
Parameter	cardNo[in]	The card number
Return	None	
Comment	None	

3.7.2.12 Request terminal risk management

Prototype	void onTermRiskManagement()	
Feature	Request terminal risk management	
Parameter	[in]	
Return	None	
Comment	None	

3.8 Tax operation module

3.8.1 Tax data exchange

Prototype	int taxDataExchange(byte[] taxSend, byte[] taxRecv)	
Feature	Exchange tax data	
Parameter	taxSend[in]	Read/write operation send data
	taxRecv[out]	Read/write operation receive data
Return	None	
Comment	Packet format for read: Header(2B)+CMD(1B)+Expect receive data length(2B,LSB) Packet format for write: Header(2B)+CMD(1B)+Data field len(2B,LSB)+Data field(LEN B) Receive packet format for read/write: Header(2B)+Response code(1B)+Data field len(2B,LSB)+Data field(LEN B) The max response data length of read/write is 1030B, following is an example of data exchange (Hex format): Read-send>>：1B 1D 09 00 06 Read-receive<<：1B 1D 00 00 06 04 01 00 02 EE DB Write-send>>：1B 1D 08 00 0B 04 06 00 11 11 03 02 0A 15 39 18 Write-receive<<：1B 1D 00 00 06 04 01 00 02 EE DB	

4. Error Code Definition

Error code	Error description
-100	Incorrect number of parameters or length
-101	Unsupported command
-1000	System parameter error
-1001	Feature not supported
-1002	Initialization failed
-1003	System time year error
-1004	System time month error
-1005	System time day error
-1006	System time hour error
-1007	System time minute error
-1008	System time second error
-1009	Hardware failure
-1010	Buffer length error
-2000	Card parameter error
-2001	Have no card
-2002	Multiple cards
-2032	Mifare card refuse CMD
-2033	Mifare card response data size not expected number

-2034	Mifare card not authenticate password
-2035	Mifare card authenticate failed
-2036	Mifare card response data error
-2037	Mifare card param illegal
-2038	Mifare Plus calculate CMAC error
-2039	Mifare Plus CMAC error
-2040	Mifare Plus AES decrypt failed
-2041	Mifare Plus AES encrypt failed
-2100	Magnetic card data decoding
-2500	Module detection failed
-2501	Drive core data structure error
-2502	Module is not powered
-2503	Carrier is not turned on
-2520	Communication timeout
-2521	Internal FIFO operation failed
-2522	Communication frame error
-2523	Communication character check error
-2524	Communication conflict
-2525	Signal in communication does not comply with the protocol
-2526	CRC check error in communication
-2527	M1 card password authentication error
-2528	Mifare authentication parameters are incorrect
-2529	Card exists
-2540	The number of data in the A card communication response does not match the expected
-2541	A card communication replies to the first character of the WUPA/REQA command is illegal
-2542	Card number checksum error of A card communication response
-2543	The first character of the card number answered by the A card communication is wrong.
-2544	A card communication response ATS TL byte is illegal
-2545	The A0 T0 byte of the A card communication response is illegal.
-2546	The A1 TA1 byte of the ATS communication is illegal.
-2547	The ATB TB1 byte of the A-card communication response is illegal.
-2548	A card communication response ATS TC1 byte is illegal
-2550	The number of data in the B card communication response does not match the expected
-2551	The first character of the B card communication response WUBP/REQB command is not 0x50
-2552	The fourth bit of the protocol type byte in ATQB is not '0'
-2553	The B-card communication responds to the difference in channel coding and setting in the ATTRIB command.
-2554	B card communication response HLTB command response non 0x00 error
-2560	Retransmitted to the limit if the correct reception is received
-2561	Block type coding error
-2562	I block PCB error or subsequent data length error
-2563	PICC uses I block response link block
-2564	Received I block serial number is incorrect
-2565	R block PCB error or subsequent data length error

-2566	PICC response NAK block
-2567	The received R block serial number is incorrect.
-2568	S block PCB error or subsequent data length error
-2569	S block non-S-WTX request sent by PICC
-2570	WTX parameter error requested (=0)
-2571	Card return data exceeds FSD
-2580	Read ID card GUID error
-2581	User canceled
-2582	MSR or IC interrupted
-2800	Verify error
-2801	Communication timeout
-2802	Module is not powered
-2803	ATR error
-2804	Communication error
-2805	PPS error
-2806	T0 param error
-2807	T0 probe byte error
-2808	T1 param error
-2809	T1 LRC error
-2810	T1 block number error
-3000	Security parameter error
-3001	Root key error
-3002	Security system is locked
-3003	Security file read and write error
-3004	Key index error
-3005	Key verification error
-3006	No PIN entry
-3007	PIN input canceled
-3008	PIN input timeout
-3009	PIN input interval is too short
-3010	KCV mode error
-3011	KCV check error
-3012	KCV ODD check error
-3013	No matching key
-3014	Error of key type
-3015	Key length error
-3016	Key exponent length error
-3017	Destination key index error
-3018	Source key index error
-3019	Source key type error
-3020	Group index error
-3022	No KCV
-3023	DUKPT overflow
-3024	DUKPT key type error

-3025	DUKPT KSN needs to add 1
-3026	Try to use the key outside the scope of the key.
-3027	Incorrect use of the key, restricting only decrypted keys to encrypt data, such as calculate the Mac with the master key
-3028	Feature not yet supported
-3029	Feature key attribute does not match
-3030	Not certified
-3031	TR31 key distribute encryption key error
-3032	TR31 key distribute MAC key error
-3033	CMAC algorithm error
-3034	Data length error
-3035	Algorithm block error
-3036	Des algorithm exception
-3037	Aes algorithm exception
-3038	Sm4 algorithm exception
-3039	Sm2 algorithm exception
-3040	Sm3 algorithm exception
-3041	Rsa algorithm exception
-3042	hash algorithm exception
-3046	POS public key exception
-3047	PAN timeout
-3048	Times exceed limit
-3049	Password error
-3050	key invalid
-3051	Not set new password
-3052	Request sensitive service
-3061	PIN/PAN anti-exhausting
-3062	The same key exists
-3081	AP key read error
-3082	AP key write error
-3083	AP key verify error
-3084	AP key lost
-3085	AP key open failed
-3086	AP key self-check failed
-3087	AP key write mode not supported
-3088	AP key invalid
-3089	AP key access timeout
-3090	AP key delete file failed
-3091	AP key other error
-4000	Transaction refuse
-4001	Please use other interfaces
-4002	Transaction terminated
-4005	Final select data error
-4100	Transaction terminated (command send and receive error)

-4101	Transaction terminated (command receipt timeout)
-4102	Transaction terminated (command receipt timeout)
-4103	Transaction terminated (status code error)
-4104	Transaction terminated (card locked)
-4105	Transaction terminated (application locked)
-4106	Transaction terminated (terminal not applied)
-4107	Transaction terminated (applications with no support for the terminal and card)
-4108	Transaction terminated (card return data error)
-4109	Transaction terminated (card return data element duplicate)
-4110	Transaction terminated (transaction not received)
-4111	Transaction terminated (card expired)
-4112	The list of preprocessing parameters is empty
-4113	Transaction terminated (L1 read card timeout)
-4114	Transaction terminated (L1 transmission error)
-4115	Transaction terminated (L1 protocol error)
-4116	Transaction terminated (L2 mandatory data error)
-4117	Transaction terminated (L2 card authentication failed (offline data authentication failed))
-4118	Transaction terminated (L2 status word error)
-4119	Transaction terminated (L2 data parsing failed)
-4120	Transaction terminated (L2 transaction amount exceeds the contactless transaction limit)
-4121	Transaction terminated (L2 card data error)
-4122	Transaction terminated (L2 does not support magnetic stripe card mode)
-4123	Transaction terminated (L2 card without PPSE)
-4124	Transaction terminated (L2 PPSE processing error)
-4125	Transaction terminated (L2 candidate list is empty)
-4126	Transaction terminated (L2 IDS read error)
-4127	Transaction terminated (L2 IDS write error)
-4128	Transaction terminated (L2 IDS data error)
-4129	Transaction terminated (L2 IDS has no matching AC)
-4130	Transaction terminated (L2 terminal data error)
-4131	Transaction terminated (L3 timeout)
-4132	Transaction terminated (L3 cancel)
-4133	Transaction terminated (L3 transaction amount does not exist)
-4134	Transaction terminated (re-presentation of the card)
-4135	Transaction terminated (using other cards (with Data Record))
-4136	Transaction terminated (using other cards)
-4137	Transaction terminated (GPO response error)
-4138	Transaction terminated (final selection of card data error)
-4139	Transaction terminated (L3 no DET data)
-4140	Kernel type is not supported
-4141	Contactless transaction limit exceeded
-4142	The amount is 0
-4144	Please use another interface (pre-processing failed)
-4500	Invalid parameter

-4501	Checksum error when downloading public key
-4502	Terminal parameters do not exist.
-4503	Error with terminal parameter data
-4504	The transaction log does not exist
-4505	Transaction log data error
-4506	EMV data does not exist
-4507	PBOC LOG format does not exist
-4825	Two present card
-4854	Complete command with empty
-4855	Complete command with ODOL
-4856	Complete command after reselect app
-4857	Read record command after reselect app
-7001	Printer error
-7002	Low battery voltage
-7003	Out of paper
-7004	Temperature is too high
-7005	Printer data error
-7006	Invalid print parameters
-7007	Device not open or device operation error
-7008	Print buffer overflow
-8001	Write data fail for tax control module
-8002	Read data fail for tax control module
-10100	Serial port closed
-10101	Serial port overtime
-10102	LRC check error
-10103	SP Out of sequence
-10104	SP is initializing
-10105	SP is rebooting
-10106	SP is reconnecting.
-10107	SP is busy
-10108	SP is sleep
-10200	Read OS file package error
-10201	SP is updating
-10202	Connect SP failed
-10203	Failed to open upgrade file
-10204	Packet timeout
-10205	Packet processing error
-10206	Upgrade string is too long
-10207	Upgrade unsuccessful
-10208	Did not get the sdk version number of this machine
-10209	The version is the same as the target upgrade version.
-10210	Query default information failed
-10211	Firmware version does not allow downgrade
-10212	Upgrade cancelled

-10300	Input parameter error
-10301	The length of the response packet data area is illegal.
-10302	Answer packet data parsing error
-10400	The kernel has been rebooted
-11000	BASE error start
-11001	Operation not permitted
-11002	No such file or directory
-11003	No such process
-11004	Interrupted system call
-11005	I/O error
-11006	No such device or address
-11007	Argument list too long
-11008	Exec format error
-11009	Bad file number
-11010	No child processes
-11011	Try again
-11012	Out of memory
-11013	Permission denied
-11014	Bad address
-11015	Block device required
-11016	Device or resource busy
-11017	File exists
-11018	Cross-device link
-11019	No such device
-11020	Not a directory
-11021	Is a directory
-11022	Invalid argument
-11023	File table overflow
-11024	Too many open files
-11025	Not a typewriter
-11026	Text file busy
-11027	File too large
-11028	No space left on device
-11029	Read-only file system
-11030	Illegal seek
-11031	Too many links
-11032	Broken pipe
-11033	Math argument out of domain of function
-11034	Math result not representable
-11107	Communication not connected
-11301	ACK response packet param error
-11302	SP ACK data overflow
-11401	CMD packet data length overflow
-11402	CMD package verify error, no info field

-11403	SP receive buffer is full, no info field
-11404	SP receive data timeout, no info field
-11406	CMD packet sequence error
-11600	CMD packet param error
-11601	Unsupported CMD packet
-11700	Firmware update failed
-11701	Firmware size exceed designed
-11702	Firmware signature verify failed
-11703	Firmware boot name error
-11704	Firmware update CMD error
-11705	Firmware update access flash error
-11706	Get device code error
-11707	SE chip type error
-20001	Repeated call
-20002	Firmware is being upgraded
-20003	Parameter error
-20004	Thread was aborted
-20005	Firmware upgrade failed
-20006	Firmware verification failed
-30001	Failure to read card (unknown cause card failure, recommended to re-read the card operation)
-30002	Unknown card type
-30003	Failure of NFC check card
-30004	Failure of IC check card
-30005	Read card timeout
-30013	This card is a chip card and can not fallback.
-30014	Create candidate list timeout
-30015	Card interaction failure
-30016	Wrong card interaction parameters
-40002	Key length error
-40003	The check value error
-40004	Store key fail
-40005	Calculate MAC error
-40006	Encryption error
-40007	Return array data length error
-40008	The MAC algorithm type not support
-40009	Length of check value error
-40010	Key index error
-40011	Decrypt error
-40012	Length of key error
-40013	Get random error
-40014	Key does not exist
-40016	Verify signature fail
-40017	Failed to get alarm information code
-40018	Key partition has run out

-40019	Inject BDK error
-50002	Transaction preprocess fail
-50003	Transaction process fail
-50004	EMV kernel process fail
-50005	PAN format error
-50006	Call PINPAD fail
-50007	None kernel data
-50008	PINPAD parameter error
-50009	EMV process not finish
-50010	The transaction type not support
-50011	Checking card information fail or timeout.
-50012	CVM error
-50013	Database operation fail
-50014	No matching CAPK
-50015	Save terminal parameter error
-50016	No matching AID
-50017	Check card fail
-50018	Call interface order error
-50019	Transaction data invalid
-50020	PIN entry cancel
-50021	PIN entry error
-50022	The index of app select error
-50023	Cert verify error
-50024	Online process error
-50025	App final select timeout
-50026	App final select error
-50027	Signature error
-50028	Unknown CVM type
-60001	Input PIN timeout
-60002	Keyboard failed to activate password
-60003	PinPadType type error (when the incoming keyboard type is not 1 and 2, the error is returned)
-60004	Getting PinBlock failed
-60005	PIN status query thread is interrupted
-70001	Miss permisstion **com.sunmi.perm.MSR**
-70002	Miss permission **com.sunmi.perm.ICC**
-70003	Miss permission **com.sunmi.perm. CONTACTLESS_CARD**
-70004	Miss permission **com.sunmi.perm. PINPAD**
-70005	Miss permission **com.sunmi.perm. SECURITY**
-70006	Miss permission **com.sunmi.perm. LED**

5. Entity class

5.1 EmvTermParamV2 – Terminal parameter entity class

Class member variable declaration:

```
public String ifDsn = "3030303030393035";  // IFD sn
public String terminalType = "22";   // Terminal type
public String countryCode = "0156";  // Terminal country code
public boolean forceOnline = false;  // Merchant force online(1-online)
public boolean getDataPIN = true;    // Whether to read the number of retries before the
password is detected?
public boolean surportPSESel = true; // Support PSE select
public boolean useTermAIPFlg = true;  // Is risk management based on card AIP?
public boolean termAIP = true; // Whether the terminal enforces risk management?
public boolean bypassAllFlg;   // After processing a PIN in byPass mode, whether other
PINs are also handled in bypass mode?
public boolean bypassPin = true; // Support bypass PIN?
public boolean batchCapture;     // Whether to batch capture data?
public boolean ectSiFlg = true;  // Does EC Terminal Support Indicator exist?
public boolean ectSiVal = true;  //Whether to support the electronic cash terminal
support indicator?
public boolean ectTlFlg = true;  //Does EC Terminal Transaction Limit exist?
public String ectTlVal = "100000"; // Electronic cash terminal transaction limit, unit
cent
public String capability = "E0F8C8";       // Terminal capability
public String addCapability = "0300C00000";// Terminal extended capability
public boolean scriptMode;                 // scriptMode
public boolean adviceFlag = true;          // adviceFlag
public boolean isSupportSM = true;         // Support SM?
public boolean isSupportTransLog = true;   // Support transaction LOG?
public boolean isSupportMultiLang = true;  // Support multiple language?
public boolean isSupportExceptFile = true; // Support exception file?
public boolean isSupportAccountSelect = true;  // Support account select?
public String TTQ = "26000080";    // Terminal transaction attribute (For contactless)
public boolean IsReadLogInCard;     // Is it an application selection process for reading
in-card transaction records?
private byte[] reserved = new byte[3];  // Reserved byte value must be 0
```

5.2 AidV2 -AID entity class

```
public byte[] aid; //AID
public byte[] cvmLmt = new byte[6]; //ContactLess CVM Limit
public byte[] termClssLmt = new byte[6]; //ContactLess Trans Limit
public byte[] termClssOfflineFloorLmt = new byte[6];  //ContactLess Floor Limit
public byte[] termOfflineFloorLmt = new byte[6]; //Terminal electronic cash transaction
limit
public byte selFlag;  //ASI
public byte targetPer;// Random target percentage
public byte maxTargetPer; // Offset the maximum target percentage randomly selected
public byte[] floorLimit; //Terminal Floor Limit
public byte randTransSel; // Whether to make random trading choices
public byte velocityCheck;// Whether to perform frequency detection
public byte[] threshold = new byte[4];// Bias randomly selected thresholds
```

```
public byte[] TACDenial = new byte[5];// TAC-Denial
public byte[] TACOnline = new byte[5];// TAC-Online
public byte[] TACDefault = new byte[5];   //TAC-Default
public byte[] AcquierId = new byte[6];// Acquirer ID
public byte[] dDOL;   //Default DDOL
public byte[] tDOL;   // Default TDOL
public byte[] version = new byte[2];  //Application version
public byte rMDLen;   // Risk management data length
public byte[] riskManData = new byte[8];  // Risk management data
public byte[] merchName = new byte[128];  //Merchant Name
public byte[] merchCateCode = new byte[2];// Merchant type code
public byte[] merchId = new byte[16]; //Merchant ID
public byte[] termId = new byte[8];   //Teminal ID
public byte[] referCurrCode = {0x01, 0x56};   //Reference currency code
public byte referCurrExp; //Reference currency code exponent
public byte[] referCurrCon = new byte[4]; // Conversion Coefficient of Reference Currency
Code and Transaction currency Code(currently, this field is ignored)
public byte clsStatusCheck; //Contactless status check
public byte zeroCheck; //zero amount check
public byte kernelType;//Kernel type(DFC10A)
public byte paramType;//Param type(DFC10B,0-default,1-contact,2-contactless)
public byte[] ttq = new byte[4]; //terminal transaction attribute 9F66
public byte[] kernelID; //kernel ID DFC10C
```

5.3 CapkV2 –CAPK entity class

```
public byte[] rid = new byte[5];// CAPK RID
public byte index;  //CAPK KeyID
public byte hashInd;// CAPK hash algorithm indicator
public byte arithInd;   //CAPK RSA algorithm indicator
public byte[] modul;   // CAPK modulus
public byte[] exponent; // CAPK Exponent
public byte[] expDate = new byte[3];// CAPK Expiry Date(YYMMDD)
public byte[] checkSum = new byte[20];  // CAPK CheckSum
```

5.4 EMVTransDataV2 - Transaction data entity class

```
public String amount;  // Transaction amount(unit: cent), must have parameter, cannot be
null or "" , when amount="0" means check the balance.

public String transType = "00"; // Transaction type default padding with "00".

public int flowType = 01; // flow type, 0x01:Standard flow ; 0x02:Simple flow ; 0x03 qPass

public int cardType = 2; //Card type, 2:IC 4:NFC
```

5.5 EMVCandidateV2 – EMV Application Candidate

```
public short index;//Index,corresponding to the priority list
public String aid;//Card Aid
public String appPreName;/App prefer name
```

```
public String appLabel;//App label
public String issDiscrData;//Data of tag 'BF0C'
public byte priority;//Priority flag
public String appName;//Local app name
public byte kernelType;//The kernel type of contactless App
```

5.6 PinPadConfigV2 - Pinpad configuration entity class

```
public class Pinpadconfig implements Parcelable {
    private int pinpadType;  // PinPad type.  0：Default Pinpad  1:Self-defined Pinpad
    private int pinType = 0;  // Pin type (0: Online PIN，1:Offline PIN)
    private boolean isOrderNumKey = false;  // true: Normal Pinpad; false: Random Pinpad
    private byte[] pan;  // Ascii format to byte. eg. "123456".getbytes("us ascii")
    private int pinkeyIndex;  // Pin Key Index
    private int maxInput = 6;  // Maximum password input (max 12 numbers)
    private int minInput = 0;  // Minimum password input
    private int timeout = 60000;  // Time out/millisecond
    private boolean isSupportbypass = true;//support bypasspin?
    private int pinblockFormat = 0;  //pinblock format,Support the following formats：
      SEC_PIN_BLK_ISO_FMT0(0)
      SEC_PIN_BLK_ISO_FMT1(1)
      SEC_PIN_BLK_ISO_FMT2(2)：If the offline transaction defaults to this format, you
                               do not need to set it.
      SEC_PIN_BLK_ISO_FMT3(3)
      //pinblock ISO-9564，Please refer to https://en.wikipedia.org/wiki/ISO_9564
     private int algorithmType = 0; //Encrypted Pin algorithm type 0-3DES (returns 8
                                    bytes), 1-SM4 (returns 16 bytes)
     private int keySystem = 0; // The key system to which Pik currently belongs is 0-
                                SEC_MKSK, 1-SEC_DUKPT. Reference Appendix: Aidl
                                Constant Key System Constant

}
```

5.7 PinPadTextConfigV2 – Pinpad showing text configuration entity class

```
public String confirm;//The confirm key text
public String inputPin;   //The input online PIN text
public String inputOfflinePin;//The input offline PIN text
public String reinputOfflinePinFormat;//The reinput offline PIN(show remain times)text
```

5.8 DrlV2 – DRL LimitSet entity class

```
public boolean isDefaultLmt = false; //Is default limitSet
public boolean statusCheck = false;  //Is enable status check
public byte zeroCheck = 1; //Is enable zero amount check, 0-online,1-not allow,2-disable
public byte[] programID; //Program ID
public byte[] cvmLmt = new byte[6];  // ContactLess CVM Limit
public byte[] termClssLmt = new byte[6]; // ContactLess Trans Limit
```

```
public byte[] termClssFloorLmt = new byte[6];//Terminal Contactless floor limit
public byte[] termFloorLmt = new byte[6];//Terminal floor limit
public boolean cvmLmtActivate = true;//Is enable CVM limit check
public boolean termClssLmtActivate = false; //Is enable terminal contactless limit check
public byte termClssFloorLmtActivate = 1;//Is enable terminal contactless floor limit,0-
disable,1-enable,2-enable but contactless floor limit not exist
```

5.9 RevocListV2– RevocationList entity class

```
public byte[] rid= new byte[5];// Application registration service provider ID
public byte index;//Key index
public byte[] sn= new byte[3];//Serial number
public byte[] reserved = new byte[3];//Reserved bytes, all bytes should be 0
```

● 6. Access permission

6.1 Permission location

The appropriate permissions should be declared in the AndroidManifest file before using each interface.

6.2 Permission definition

6.2.1 Magnetic stripe card permission

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

6.2.2 Contact IC card permission

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

6.2.3 Contactless IC card permission

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

6.2.4 Pinpad permission

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

6.2.5 Security permission

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

6.2.6 LED permission

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

6.2.7 Printer permission (not supported)

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

6.2.8 Serial port permission (not supported)

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

6.2.9 Customer display permission (not supported)

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

6.2.10 ID card permission (not supported)

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

6.2.11 Money box permission (not supported)

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

6.2.12 Finger print permission (not supported)

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

7. Appendix

7.1 Aidl constants class

(com.sunmi.pay.hardware.aidl.AidlConstants)

7.1.1 Card type constant definition

```
// Magnetic
public static final int MAGNETIC= 1<<0;
// IC
public static final int IC = 1<<1;
// RFC
public static final int NFC = 1<<2;
// Mifare
public static final int MIFARE= 1<<3;
// PSAM,slot 0
public static final int PSAM0= 1<<4;
// Felica
public static final int FELICA= 1<<5;
// SAM1
public static final int SAM1= 1<<6;
// Mifare plus
public static final int MIFARE_PLUS= 1<<7;
// Mifare desfire
public static final int MIFARE_DESFIRE= 1<<8;
// AT24C01
public static final int AT24C01= 1<<9;
// AT24C02
public static final int AT24C02= 1<<10;
// AT24C04
public static final int AT24C04= 1<<11;
// AT24C08
public static final int AT24C08= 1<<12;
// AT24C16
public static final int AT24C16= 1<<13;
// AT24C32
public static final int AT24C32= 1<<14;
// AT24C64
public static final int AT24C64= 1<<15;
// AT24C128
public static final int AT24C128= 1<<16;
// AT24C256
public static final int AT24C256= 1<<17;
// AT24C512
public static final int AT24C512= 1<<18;
// SLE4442
public static final int SLE4442= 1<<19;
// SLE4428
public static final int SLE4428= 1<<20;
// AT88SC1608
public static final int AT88SC1608= 1<<21;
// CTX512B
public static final int CTX512B= 1<<22;
```

7.1.2 Key type constant definition

```
// KEK (Key encrypt key)
public final static int KEY_TYPE_KEK = 0x01;
// TMK (Terminal master key)
public final static int KEY_TYPE_TMK = 0x02;
// PIK (PIN key)
public final static int KEY_TYPE_PIK = 0x03;
// MAK (Mac key)
public final static int KEY_TYPE_MAK = 0x04;
// TDK (Track data key)
public final static int KEY_TYPE_TDK = 0x05;
// Reserved
public final static int KEY_TYPE_REC = 0x06;
// Dupkt BDK (Base derived key)
public static final int KEY_TYPE_DUPKT_BDK = 0x07;
// Dukpt IPEK (Initial PIN encryption key)
public static final int KEY_TYPE_DUPKT_IPEK = 0x08;
// TR31 KBPK (Key block protection key)
public static final int KEY_TYPE_KBPK = 0x09;
// Account data key
public static final int KEY_TYPE_TADK =0x0A;
```

7.1.3 Key algorithm type constant definition

```
// Encryption type: 3DES or DES
public final static int KEY_ALG_TYPE_3DES = 0x01;
// Encryption type: AES
public final static int KEY_ALG_TYPE_AES = 0x02;
// Encryption type: SM4
public final static int KEY_ALG_TYPE_SM4 = 0x03;
```

7.1.4 MAC algorithm constant definition

```
// ISO9797-1-ALG1
public static final int MAC_ALG_ISO_9797_1_MAC_ALG1 = 1001;
// ISO9797-1-ALG3
public static final int MAC_ALG_ISO_9797_1_MAC_ALG3 = 1003;
 // Same as ISO9797-1-ALG3
public static final int MAC_ALG_ISO_16609_MAC_ALG1 = 2000;
// ECB algorithm.
public static final int MAC_ALG_FAST_MODE = 3000;
// X9_19 algorithm mac
public static final int MAC_ALG_X9_19 = 3001;
// CBC UnionPay algorithm
public static final int MAC_ALG_CBC = 3002;
// FAST_MODE internation standard mac
public static final int MAC_ALG_FAST_MODE_INTERNATIONAL = 30000;
// CBC international standard mac
public static final int MAC_ALG_CBC_INTERNATIONAL = 30001;
public static final int MAC_ALG_CUP_SM4_MAC_ALG2 = 3004;
```

7.1.5 DukptKeyTpe constant definition

```
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 Key system type constant definition

```
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 Encryption mode constant definition

```
public static final int DATA_MODE_ECB = 0;
public static final int DATA_MODE_CBC = 1;
public static final int DATA_MODE_OFB = 2;
public static final int DATA_MODE_CFB = 3;
```

7.1.8 Dukpt key select constant definition

```
// DUKPT PIN key
public static final int DUKPT_KEY_SELECT_KEY_PIN = 0;
// DUKPT request and response Mac key
public static final int DUKPT_KEY_SELECT_KEY_MAC_BOTH = 1;
// DUKPT response Mac key
public static final int DUKPT_KEY_SELECT_KEY_MAC_RSP = 2;
// DUKPT request and response data key
public static final int DUKPT_KEY_SELECT_KEY_DATA_BOTH = 3;
// DUKPT response data key
public static final int DUKPT_KEY_SELECT_KEY_DATA_RSP = 4;
// DUKPT calculate Mac key (dukpt-aes)
public static final int DUKPT_KEY_SELECT_KEY_MAC_GEN = 5;
// DUKPT data encrypt key (dukpt-aes)
public static final int DUKPT_KEY_SELECT_KEY_DATA_ENC = 6;
// DUKPT 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 constant definition

```
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 signature algorithm constant definition

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

7.1.11 Hash type constant definition

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

7.1.12 Certificate type constant definition

```
// ID card
public static final int IDCARD = 536911872;
// Certificate of officers
public static final int ARMYCARD = 536911873;
// Passport
public static final int PASSPORT = 536911874;
// Arrival card
public static final int ARRIVALCARD = 536911875;
// Temporary ID card
public static final int TEMPIDCARD = 536911876;
// Other certificate
public static final int OTHERCARD = 536911877;
```

7.1.13 EMV module related constant definition

```
// Force online
public static final int FORCE_ONLINE = 0;
// No online
public static final int NO_ONLINE = 1;
// CAPK and AID are not exist.
public static final int EXIST_ALL_NOT = -1;
// CAPK and AID are exist.
public static final int EXIST_ALL= 0;
// CAPK is not exist
```

```
public static final int EXIST_CAPK_NOT = 1;
// AID is not exist.
public static final int EXIST_AID_NOT = 2;
// Transaction complete.
public static final int EMV_RESULT_FINISHED = 0x9000;
// Transaction abort.
public static final int EMV_RESULT_TERMINATION = 0x9001;
// Get PINBLOCK failure.
public static final int EMV_ERROR_PINBLOCK = 0x9002;
// Transaction is not supported.
public static final int EMV_UNSUPPORTED_TRANS = 0X9003;
```

7.1.14 EMV FlowType constant definition

```
// Sandard authorization flow
public static final int TYPE_EMV_STANDARD = 0x01;
// Simple flow, finished as soon as read card number
public static final int TYPE_EMV_BRIEF = 0x02;
// QPASS flow-Skip input PIN in contactless transaction
public static final int TYPE_NFC_SKIP_CVM = 0x03;
// Contactless speedup flow
public static final int TYPE_NFC_SPEEDUP = 0x04;
```

7.1.15 EMV clean data constant definition

```
// Clear all data
public static final int OP_CLEAR_DATA_ALL = 0;
// Clear terminal data
public static final int OP_CLEAR_DATA_TERMINAL = 1;
// Clear card data
public static final int OP_CLEAR_DATA_CARD = 2;
```

7.1.16 EMV TLV operation type constant definition

```
// 普通
public static final int OP_NORMAL = 0;
// PayPass
public static final int OP_PAYPASS = 1;
// PayWave
public static final int OP_PAYWAVE = 2;
// MIR
public static final int OP_MIR = 3;
// PAGO
public static final int OP_PAGO = 4;
// JCB
public static final int OP_JCB = 5;
// PURE
public static final int OP_PURE = 6;
// AE
public static final int OP_AE = 7;
// FLASH
public static final int OP_FLASH = 8;
// DPAS
public static final int OP_DPAS = 9;
```

```
// RUPAY
public static final int OP_RUPAY = 10;
// EFTPOS
public static final int OP_EFTPOS = 11;
// AID RELEVANT
public static final int OP_AID_RELEVANT = 101;
// Add customized tag
public static final int OP_ADD_SELF_DEFINE_TAG = 102;
// Delete customized tag
public static final int OP_DEL_SELF_DEFINE_TAG = 103;
```

7.1.17 EMV kernel type definition

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

7.1.18 EMV param type definition

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

7.1.19 EMV transaction result code definition

```
// Success
public static final int SUCCESS = 0;
// offline approval
```

```
public static final int OFFLINE_APPROVAL = 1;
// offline denied
public static final int OFFLINE_DECLINE = 2;
// Reserved
public static final int RESERVE = 3;
// Retap
public static final int TRY_AGAIN = 4;
// online approval
public static final int ONLINE_APPROVAL = 5;
// online denied
public static final int ONLINE_DECLINE = 6;
```

7.1.20 AID function behavior constant definition

```
// Add or update a AID.
public static final int ACTION_AID_ADD = 0x00;
// Delete all AIDs.
public static final int ACTION_AID_DEL = 0x01;
```

7.1.21 CAPK function behavior constant definition

```
// Add or update a CAPK.
public static final int ACTION_CAPK_ADD = 0x00;
// Delete all CAPKs.
public static final int ACTION_CAPK_DEL = 0x01;
```

7.1.22 System parameter constant definition

```
// Hardware version
public static final String HARDWARE_VERSION = "HardwareVersion";
// Firmware version
public static final String FIRMWARE_VERSION = "FirmwareVersion";
// SN
public static final String SN = "SN";
// PN
public static final String PN = "PN";
// TUSN
public static final String TUSN = "TUSN";
// Device code(eg.W6900(4G))
public static final String DEVICE_CODE = "DeviceCode";
// Device model(eg.P1N)
public static final String DEVICE_MODEL = "DeviceModel";
// Reserved. (value is Json format- extensible)
public static final String RESERVED = "Reserved";
// PIN input mode
public static final String PINPAD_MODE = "PinPadMode";
// Contactless param for A card
public static final String PCD_PARAM_A = "PCD_PARAM_A";
// Contactless param for B card
public static final String PCD_PARAM_B = "PCD_PARAM_B";
// Contactless param for Felica card
public static final String PCD_PARAM_C = "PCD_PARAM_C";
// Key same check flag
public static final String SEC_MODE = "SecMode";
```

```
// IC driver version
public static final String PCD_IFM_VERSION = "PCD_IFMVersion";

// EMV version
public static final String EMV_VERSION = "EMVVersion";
// Paypass version
public static final String PAYPASS_VERSION = "PaypassVersion";
// Paywave version
public static final String PAYWAVE_VERSION = "PaywaveVersion";
// QPBOC version
public static final String QPBOC_VERSION = "QPBOCVersion";
// Entry version
public static final String ENTRY_VERSION = "EntryVersion";
// Mir version
public static final String MIR_VERSION = "MirVersion";
// JCB version
public static final String JCB_VERSION = "JCBVersion";
// Pago version
public static final String PAGO_VERSION = "PAGOVersion";
// PURE version
public static final String JCB_VERSION = "PUREVersion";
// AE version
public static final String PAGO_VERSION = "AEVersion";
//FLASH version
public static final String FLASH_VERSION = "FLASHVersion";
// DPA Sversion
public static final String DPAS_VERSION = "DPASVersion";
// APEMV version
public static final String APEMV_VERSION = "APEMVVersion";
//EFTPOS version
public static final String EFTPOS_VERSION = "EFTPOSVersion";
// EMV kernel checksum
public static final String EMV_KERNEL_CHECKSUM = "EmvKernelCheckSum";
// Pure full version
public static final String PURE_VERSION_FULL = "PUREVersionFull";
// EFTPOS full version
public static final String EFTPOS_VERSION_FULL = "EFTPOSVersionFull";
// APEMV full version
public static final String APEMV_VERSION_FULL = "APEMVVersionFull";
```

7.1.23 Led constant definition

```
// Red light
public static final int RED_LIGHT = 1;
// Green light
public static final int GREEN_LIGHT = 2;
// Yellow light
public static final int YELLOW_LIGHT = 3;
// Blue light
public static final int BLUE_LIGHT = 4;
```

7.1.24 PinPad mode constant definition

```
// Normal
public static final String MODE_NORMAL = "Normal";
// MeiTuan
```

```
public static final String MODE_MEITUAN = "MeiTuan";
// Silent
public static final String MODE_SILENT = "Silent";
```

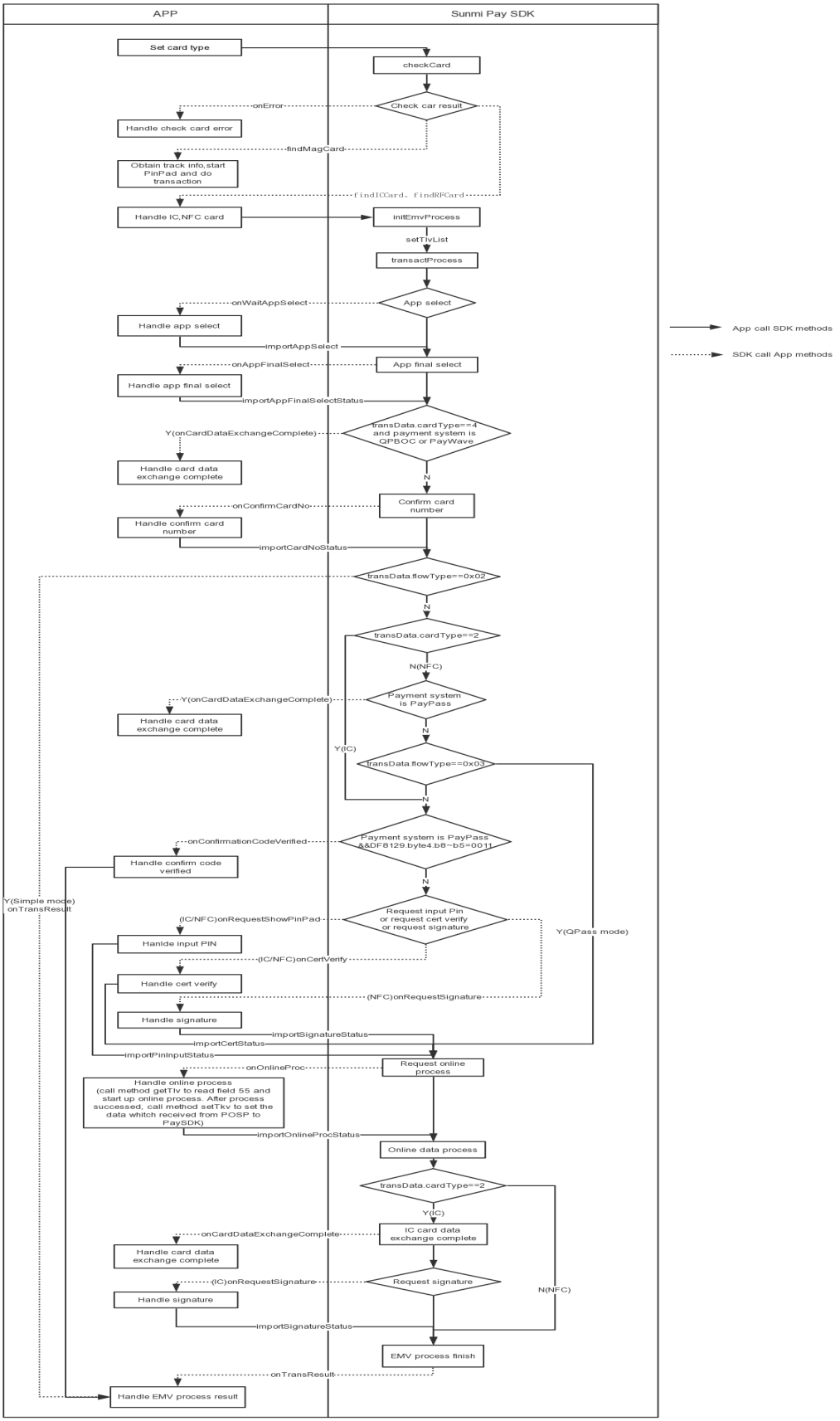
7.1.25 PinBlock format constant definition

```
public static final int SEC_PIN_BLK_ISO_FMT0 = 0;
public static final int SEC_PIN_BLK_ISO_FMT1 = 1;
public static final int SEC_PIN_BLK_ISO_FMT3 = 3;
 public static final int SEC_PIN_BLK_ISO_FMT4 = 7;
```

7.2 PAN data interception

Starting from the second digit on the right side of the card number (Field 2), the 12 bit is taken to the left as the PAN to participate in PIN encryption and decryption.
For example, the card number is 6225882145611077.
PAN data after interception 588214561107
Convert to byte[] array "588214561107".GetBytes ("US-ASCII");
All the PAN data that needs to be passed in the SDK is intercepted in this way, and if the pinblock result is incorrect, check the incoming PAN data in accordance with the above rules if the pinKey is correct

7.3 EMV trans flow chart



7.4 Key Index illustration

7.4.1 Overview

SP(SecureProcessor) provide 200 MKSK (3DES/AES/SM4) key indices, 10 Dukpt-3DES key indices, 10 Dukpt-AES key indices, 100 Dukpt-3DES extension key indices. The following table describle the mapping rule between key type and SP key index, and how **SunmiPayHardwareService(SPHS)** to control the access of each type key:

Key system	Algorithm	SP key index range	Access control rule of SPHS
MKSK	3DES/AES/SM4	1-200	Using **KeyPartition**: Each KeyPartion belong to one clinet APP, and can only be accessed by this App. App’ s package name and developer signature will be checkd when it accessing KeyPartition. Other App cannot access this App’ s KeyPartition.
DUKPT	3DES	1-10	--
DUKPT	AES	11-20	--
DUKPT	3DES-extenstion	1101-1200	Using **key exclusive mechanism** ：Each key has a full mapping rule with SP, eg: client App1 Save key at index 1100➔Mapping to SP 1101, App2 save key at 1100➔Mapping to SP 1102. The delete and recycle rule of key is similar to **KeyPartition** recycle rule

7.4.2 Support KeyPartition device model

Device model	Is support KeyPartiton by default	Can open KeyPartition by code
P1N/P1_4G	No	Yes
P2lite	Yes	Yes
P2/P2_Pro	Yes	Yes
P2Mini	Yes	Yes

7.4.3 SPSH control rule of KeyPartition

SunmiPayHardwareService(SPHS) logically divide 200 个 MKSK key indices to 10 KeyPartitions, The first dividend KeyPartition is used by SPHS itself, and SPSH provide the rest 9 KeyParitions for client App. The key index range of each KeyPartition is 0~19. Client App use this ranged keyIndex to save/use(encryption/decryption/calculate Mac)/delete key. After client App uninstalled, if other new client App save key and both all the 9 KeyPartitions are distributed, the uninstalled KeyPartition will be recycled, it stored KeyPartition info will be erased.

7.4.4 Key index mapping rule

SP(SecureProcessor) key index start from 1, App' s key index mapping rule is:

SunmiPayHardwareService(SPHS) (0-19)➔ SP(1-20)

Client App 1（0~19）➔ SP(21~40)

Client App 2（0~19）➔ SP(41~60)

……

Client App 9（0~19）➔ SP(181~200)