WPOS SDK Development Reference Part I

V-1.4.8

Version control information and description

Version No	Date	Writer	Description
V-1.0	2018.1.15	SDK Team	
V-1.1	2018.1.20	SDK Team	Add to check card during printing
V-1.2	2018.1.24	SDK Team	Add DesFirecard interface
V-1.3	2018.2.26	SDK Team	Add PIN negative return code
V-1.3.1	2018.3.19	SDK Team	English version review
V-1.3.2	2018.5.11	SDK Team	Add to set kernel
V-1.3.3	2018.5.25	SDK Team	Add function: Get the non-connected authentication version Get contact authentication version Get the device PCI authentication version Remaining number of logical contact card passwords Logical contact card to modify password
V-1.3.4	2018.6.21	SDK Team	Add getKeyKCV function Modify the description for the getKSN function
V-1.3.5	2019.1.24	SDK Team	Add two printPDF417 function(session 8.5.4 and 8.5.5) Modify the initBlueToothPrint function(add new value for Parameter Input: 3: off-line 4: print object not connected 10: print finish add new Result values:) Add return values for startBlueToothPrint function(add new return value: 5: exception occurs)
V-1.3.6	2019.1.28	SDK Team	Add Scan code functions(Chapter 10): 10.1 init 10.2 Close Scan module 10.3 Single Scan Code 10.4 Multiple Scan Code 10.5 Consecutive single scan 10.6 Cancel code sweep operation 10.7 Maximum number of multiple sweeps 10.8 Set lighting mode Add return values for Scan code module Add Logic_I2C Function(session 2.33) Add InjectIPEKByKEK Function(session 3.2) Add SetGrayLevel Function(session 8.3)
V-1.3.7	2019.7.12	SDK Team	Add TR-31 injectKeyBlock

V-1.3.8	2019.8.15	SDK Team	Add function Get Device POST State(session 1.18)
V-1.3.9	2019.9.03	SDK Team	Add Cortex Scan Module(session 12)
V-1.4.0	2019.9.11	SDK Team	Add function setPrintType(session 9.9)
V-1.4.1	2019.9.17	SDK Team	Add function setPrintPaperType(session 9.10)
V-1.4.2	2019.10.11	SDK Team	Add function startPinInputOff(session 6.2.2)
V-1.4.3	2019.11.25	SDK Team	Add function printQRCode(add margin Param) (session 9.5.6)
V-1.4.4	2019.12.27	SDK Team	Update decode_SetMaxMultiReadCount(session 11.7)
V-1.4.5	2020.1.13	SDK Team	Add function getOfflinePINTryCounter(session 6.2.4) Add function startPinInputOffLineForSingle(session 6.2.3)
V-1.4.6	2020.3.18	SDK Team	Add function setPrintLineSpacing(session 9.11)
V-1.4.7	2020.11.30	SDK Team	Add RSA API(session 13)
V-1.4.8	2020.0203	SDK Team	Add printMultiseriateString

Document Overview

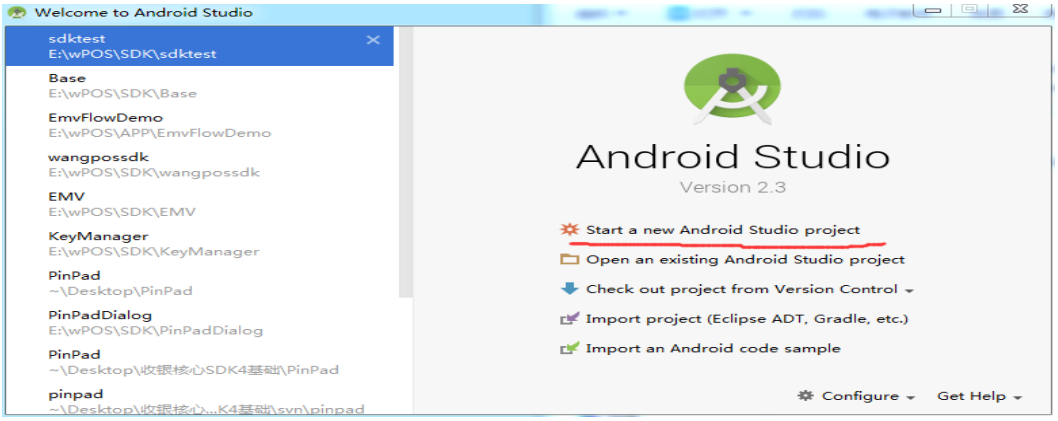
This document, *SDK Development Reference* I, based on the WPOS platform devices, which provide the capability and convenience for customers to understand the WPOS SDK, independently develop Android applications, and quickly integrate the development work.

Development Environment Setup

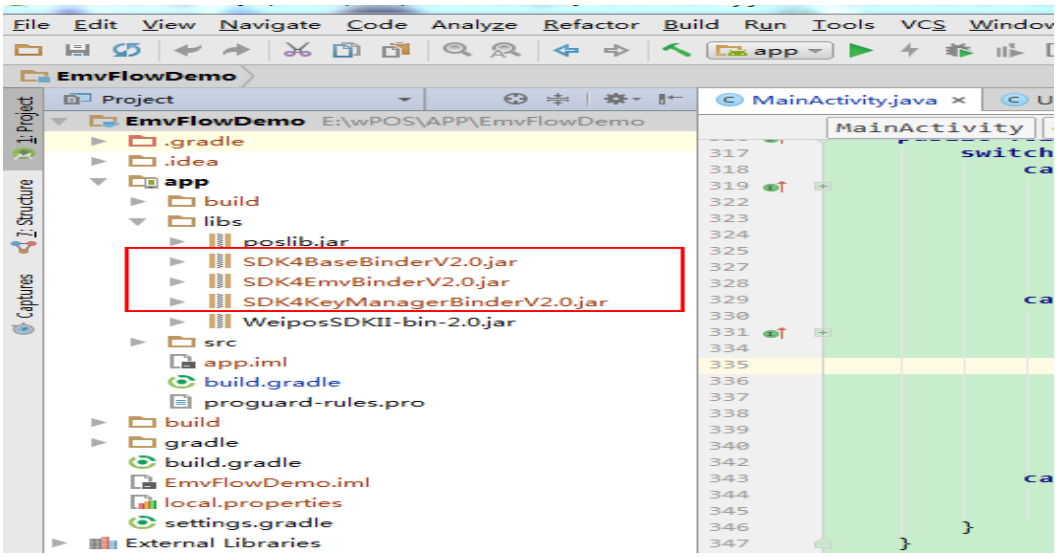
Step1: Download and Install Android Studio

You can download Android Studio from the official Website,
On the official website also have a reference for the installation

Step2: Start a new WPOS SDK project



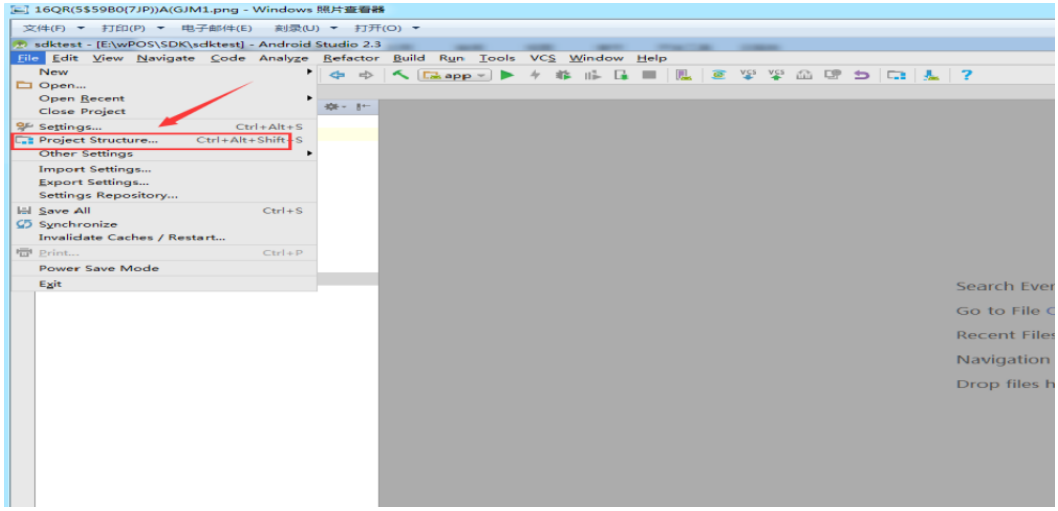
Copy WPOS SDK4's jar file to the project libs Directory, as shown in screenshot below:



Operation steps:

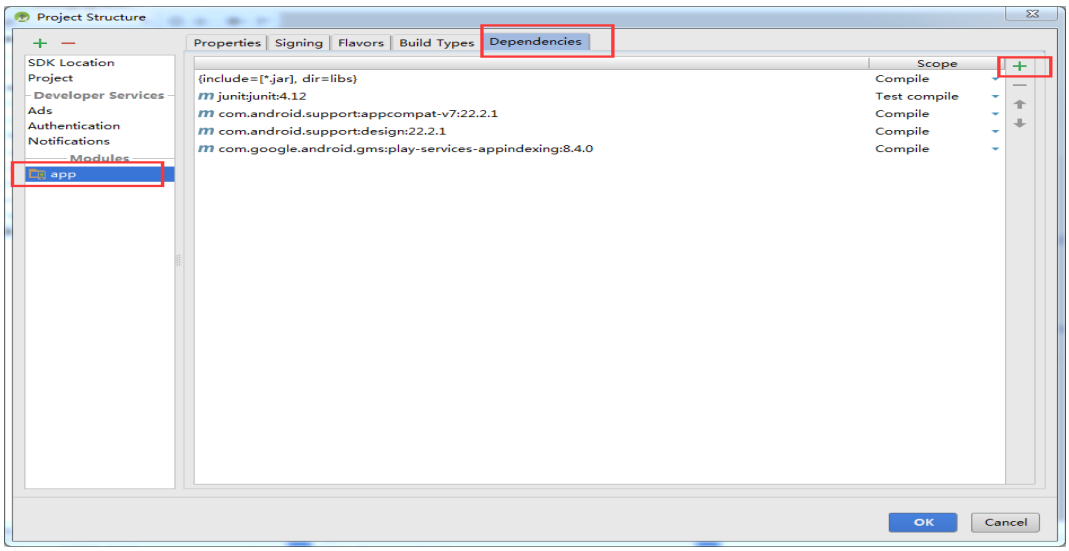
1.File->Project Structure

Select the File->Project Structure on the menu bar



2. Find out the module label in the pop-out dialog,

select your module, then select tab-page Dependencies



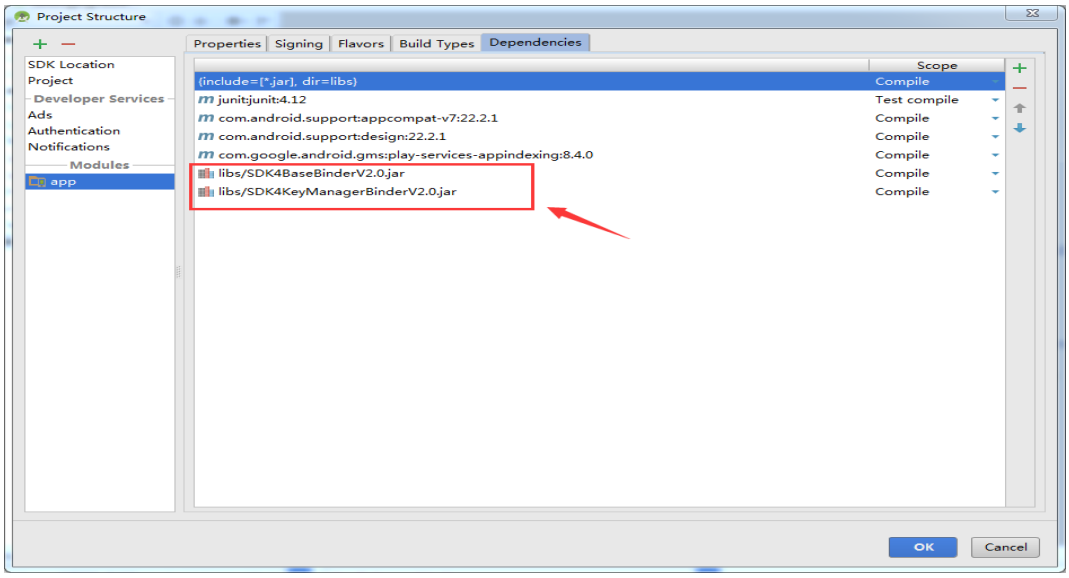
3. Click “+” button, select the jar files in lib directory, including:

   SDK4BaseBinderVx.x.jar

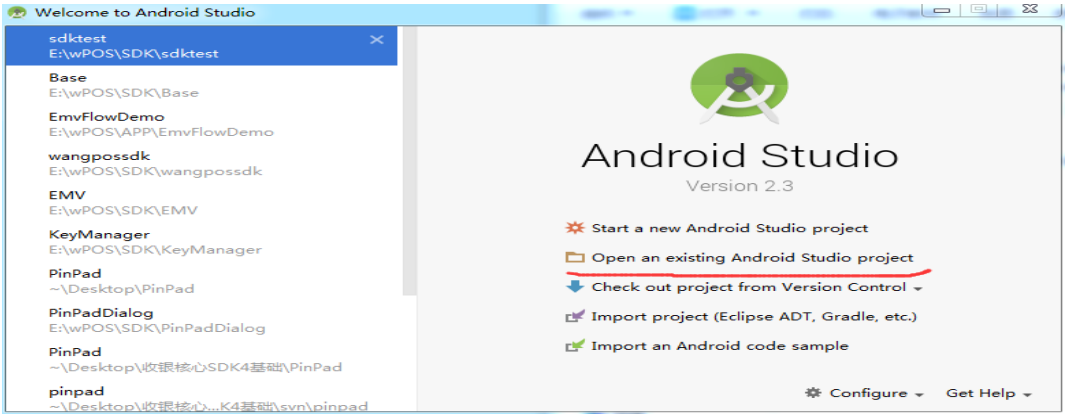
   SDK4EmvBinderVx.x.jar

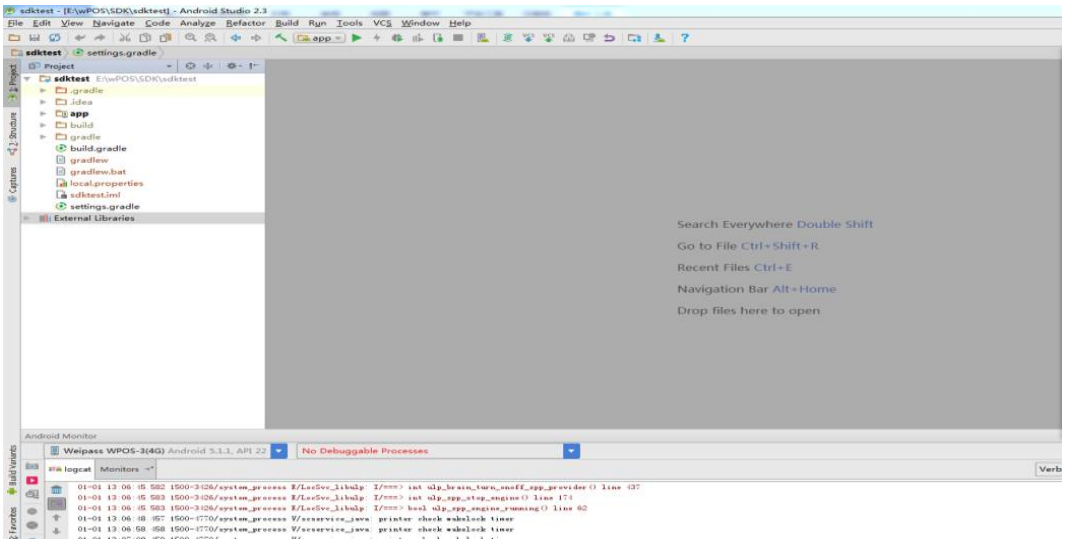
   SDK4KeyManagerBinderVx.x.jar

   Finish adding, the dependency information will Synchronize to the module’s build.gradle



Step3: Import SDK Demo Project

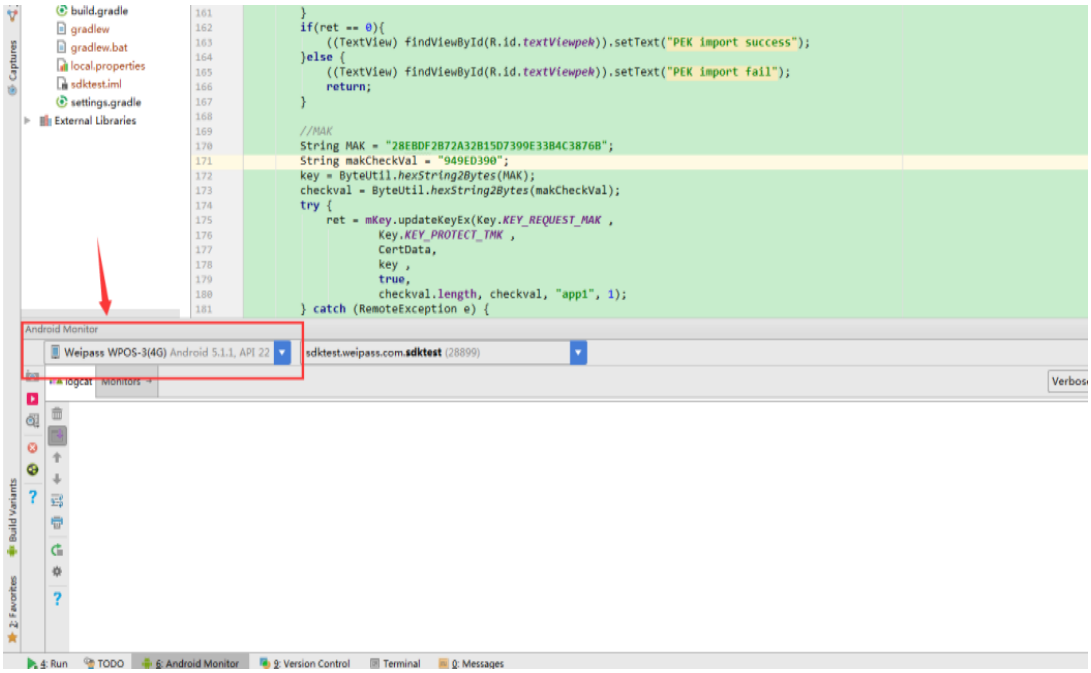




Step4: Connect to the WPOS device

You shall connect to the WPOS device with the USB cable, and the POS shall activate the Developer Mode.

When the android studio detects the connected Devices, as shown in screenshot below:



Step5: Run Project

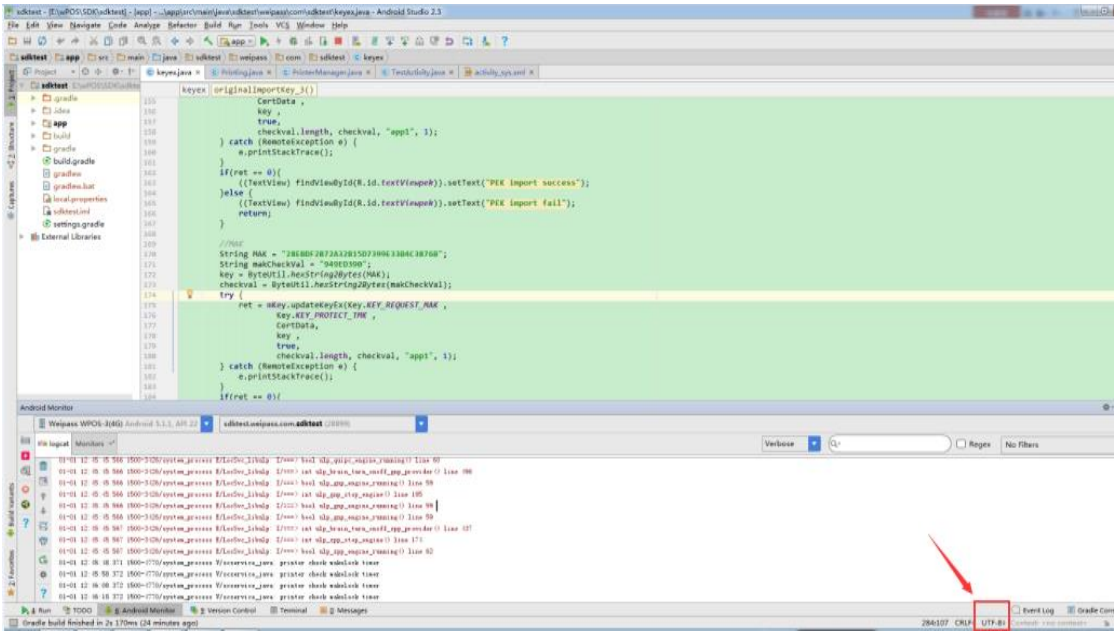
Click the 

to run the project on your device.

Development Need to Know

1. Android Studio with UTF-8

The default encode for the Android Studio is UTF-8, as shown in screenshot below:



2. Take logs:

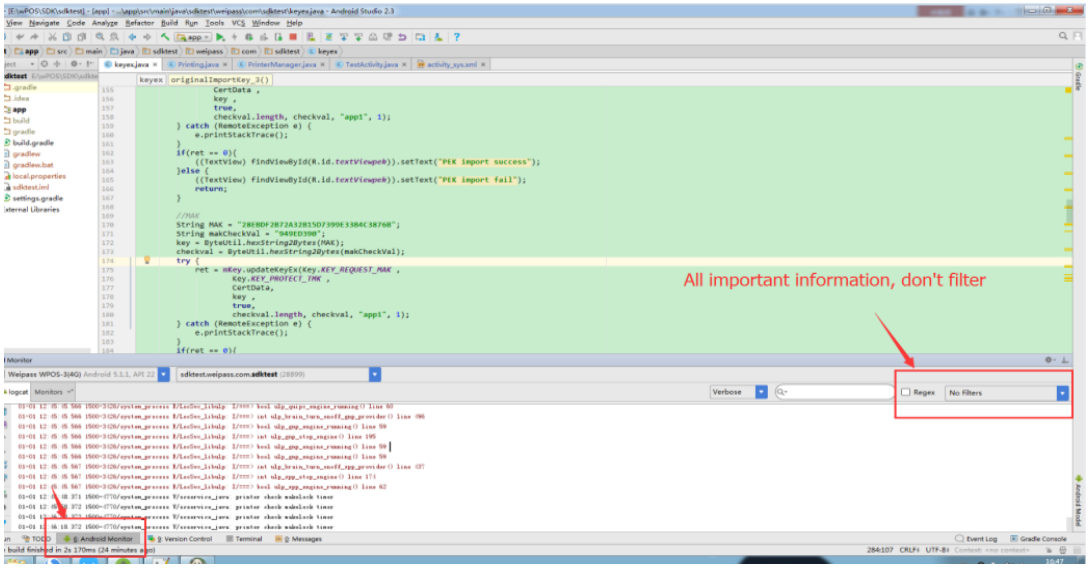
If you have any problems when you using WPOS SDK, please take the logcat and feedback to WPOS engineer team.

You can follow these steps:

1) Select Android Monitor in the bottom tool-bar in Android Studio.

In the extension dialog, select the Tab logcat, un-click the check-box Regex

select No Filter in the drop-down list beside the check-box Regex



2) Copy the log-information in the logcat textbox, save as txt type file, and give feedback to WPOS engineer team.

Compatibility: These Step is Compatible for all types WPOS devices.

1. System Function

Note:

You shall new a Class Core Instance, before calling the System Function API.
And the instantiation shall be process in a Thread routine,
as shown the code segment below:

```
new Thread(new Runnable() {
        @Override
        public void run() {
                mCore = new Core(getApplicationContext());
        }
}).start();
```

1.1 Get SP Date/Time

Function Name	**getDateTime**
Parameter Input	N/A
Parameter Output	byte[] datetime = new byte[14]; Response Data
Result	int result; 0:Success Others: Not success
Remarks	String strDate = **new** String(datetime);

1.2 Set SP Date/Time

Function Name	**setDateTime**
Parameter Input	byte[] datetime Set Date/Time (yyyyMMDDhhmmss)
Parameter Output	N/A
Result	int result; 0:Success Others: Not success
Note	1971 < year < 2099;
Remarks	String str = **"20171212010101"**; datetime = str.getBytes(**"UTF-8"**);

1.3 Get Device SN Serial Number

Function Name	Android SDK Version >= 29 String deviceSN = Build.SERIAL; Android SDK Version >= 27 String deviceSN = Build.getSerial(); Android SDK Version < 27 String deviceSN = Build.SERIAL;

1.4 Write SP SN Serial Number

Function Name	**writeSN**

Parameter Input	byte[] sN = snStr.getBytes("UTF-8"); int length = sN.length();
Parameter Output	N/A
Result	int result; 0:Success Others: Not success

1.5 Read SP SN Serial Number

Function Name	readSN
Parameter Input	N/A
Parameter Output	byte[] sN = new byte[32]; int[] length = new int[1];
Result	int result; 0:Success Others: Not success
Remarks	String strSN = **new** String(sN); Note: Need to write SP SN number before reading SP SN number

1.6 Get Battery Level

Function Name	getBatteryLevel
Parameter Input	N/A
Parameter Output	byte[] level = new byte[3];
Result	int result; 0:Success Others: Not success
Remarks	String val = **new** String(level); val = val.substring(0,1) + "." + val.substring(1,3) + "v";

1.7 Get Tamper State

Function Name	getTamper
Parameter Input	N/A
Parameter Output	byte[] data = new byte[2]; int[] length = new int[1];
Result	int result; 0:Success Others: Not success
Remarks	data[0] : 0x00: The security mechanism of SP is inactive and has not been triggered (At the point, the ***data[1]*** means tamper-alarm connected, but inactive), this moment the tamper-chip is in initial state. 0x01: The security mechanism of SP is active. The device state relies on ***data[1]***. If ***data[1]*** is non-zero, which means the device tamper-alarm is active, the **app** shall remind to reset the security module. Otherwise means the device work normally. 0x02:

	The security mechanism of SP is not active (the device has been reset the security module. Device needs to be active via the authorization mechanism). data[1] : Return the state for temper: - The Normal State: The higher 4 bits shall be set to 0, and the lower 4 bits indicate the status of the temper connections. If one of the lower 4 bits set to 1, means one of the temper-connection disconnected. So, 0x06(0000,**0110**) means T1, T2 connection disconnect, and T0, T3 connection is fine. - The exception State: The higher 4 bits shall set to 1, for example: 0xF2(**1111,**0010) means internal checking Failed, shall do it again.

1.8 Enable Tamper

Function Name	**enableTamper**
Parameter Input	N/A
Parameter Output	byte[] data = new byte[1]; int[] length = new int[1];
Result	int result; 0:Success Others:Not success
Remarks	data[0]: 0x00 -> Active Success 0xF1 -> The temper isn't ready 0xF2 -> Security mechanism initializing failed 0xF3 -> Dual-control Key hasn't been injected. 0xF4 -> The public key for updating hasn't been injected. Other: Error

1.9 Get Device Build Version

Function Name	String deviceVersion = Build.*DISPLAY*;

1.10 Get Device SP Version

Function Name	**getDevicesVersion**
Parameter Input	N/A
Parameter Output	byte[] data = new byte[128]; int[] length = new int[1];
Result	int result; 0:Success Others: Not success

Remarks	String devVersion = **new** String(data);

1.11 Get Device SP ID

Function Name	**getSpID**
Parameter Input	N/A
Parameter Output	byte[] id = new byte[64]; int[] idLen = new int[1];
Result	int result; 0:Success Others: Not success

1.12 Buzzer

1.12.1 Buzzer

Function Name	**buzzer**
Parameter Input	N/A
Parameter Output	N/A
Result	int result; 0:Success Others: Not success

1.12.2 Buzzer for long

Function Name	**buzzerEx**
Parameter Input	int time(ms) Set Buzzer duration
Parameter Output	N/A
Result	int result; 0:Success Others: Not success

1.13 Device LED

1.13.1 LED on

Function Name	**led**
Parameter Input	int blue 1:On 0:Off int yellow 1:On 0:Off int green 1:On 0:Off int red 1:On 0:Off int open 1:Open 0:Close
Parameter Output	N/A
Result	int result; 0:Success Others: Not success

1.13.2 LED Flash

Function Name	**ledFlash**		
Parameter Input	int	blue	1:On 0:Off
	int	yellow	1:On 0:Off
	int	green	1:On 0:Off
	int	red	1:On 0:Off
	int	openTime (ms)	On Duration
	int	closeTime (ms)	Off Duration
	int	duration (ms)	Flash Duration
Parameter Output	N/A		
Result	int	result;	0:Success Others: Not success

1.14 SetKernel

Function Name	**SetKernel**		
Parameter Input	int	aParam	reserved
	int	sKernel	Bit6:Whether the cardholder authentication of non-connected transactions is initiated by the kernel, that is, | 0x40, is initiated by the application by default Bit5:Setting this flag, the kernel does not directly initiate online pin input, but informs the application,, that is, | 0x20, which is not set by default Bit4~3:set the kernel type, 0:QPBOC 1:paywave
	int	bParam	reserved
Parameter Output	N/A		
Result	int	result;	0:Success Others: Not success

1.15 CLGetVersion

Function Name	**CLGetVersion**
Parameter Input	N/A
Parameter Output	byte[] version = new byte[128]; int[] verLen = new int[1];

	byte[] checkSum = new byte[128]; int[] checkLen = new int[1];
Result	int result; 0:Success Others: Not success

1.16 ScrdGetVersion

Function Name	ScrdGetVersion
Parameter Input	N/A

Parameter Output	byte[] version = new byte[128];
	int[] verLen = new int[1];
	byte[] checkSum = new byte[128];
	int[] checkLen = new int[1];

Result	int result; 0:Success Others: Not success

1.17 getFirmWareNumber

Function Name	getFirmWareNumber
Parameter Input	N/A
Parameter Output	N/A
Result	String result;

1.18 Get Device POST State

Function Name	getDeviceStatus
Parameter Input	N/A
Parameter Output	byte[] data = new byte[128]; int[] length = new int[1];
Result	int result; 0:Success Others: Not success
Remarks	data[0]: Magnetic stripe Card Reader 0x00: Normal; 0x01: Abnormal 0xFF: does not exist data[1]: Contact IC Card Reader 0x00: Normal; 0x01: Abnormal 0xFF: does not exist data[2]: Contactless IC Card Reader 0x00: Normal; 0x01: Abnormal 0xFF: does not exist data[3]: Print Module 0x00: Normal; 0x01: Abnormal 0x10: out of paper 0xFF: does not exist data[4]: PIN Module

	0x00: Normal； 0x01: Abnormal 0xFF: does not exist data[5]： Security Module 0x00: Normal； 0xFF: does not exist 0x01:External Tamper trigger 0x02:SHIELD trigger 0x03:Low temperature trigger 0x04:High temperature trigger 0x05:Low Voltage trigger 0x06:High Voltage trigger 0x0A:Tamper trigger status is not cleared 0x11:Firmware verification failed 0x12:Update public key verification failed 0x13:Key area verification failed Data[6]: Reserved Data[7]: Reserved

2. Card Operation

Note:

You shall new a Class BankCard Instance, before calling the Card Operation API.
And the instantiation shall be process in a Thread routine,
as shown the code segment below:

```
new Thread(new Runnable() {
    @Override
    public void run() {
        mBankCard = new BankCard(getApplicationContext());
    }
}).start();
```

2.1 Break Off Command

Function Name	**breakOffCommand**
Parameter Input	N/A
Parameter Output	N/A
Result	int result; 0:Success Others: Not success
Notes	The API is used to force-interrupt the last Operation. Use for: 1.The last operation doesn't have a normal-return.

	2.Once a transaction complete.

2.2 Open/Close Card Reader

Function Name	**openCloseCardReader**
Parameter Input	int readerType; Card Reader Type 0x01: Contact IC Card PSAM: 0x41: PSAM1 0x81: PSAM2 If the PSAM card is deal with the Public Transport Card (for example in China): 0x51: PSAM Card 1 0x91: PSAM Card 2 0x02: Contactless IC Card 0x04: Magnetic stripe card int operation; Operation Type 0x01: Open 0x02: Close
Parameter Output	N/A
Result	int result; 0:Success Others: Not success

2.3 Card Read

2.3.1 ReadCard (Unblock-mode)

Function Name	**cardReaderDetact**
Parameter Input	int cardtype; 0x00:Bank Card 0x01:Industry Card int readertype 0x01:Contact IC Card 0x02:Contactless IC Card 0x04:Magnetic stripe card In addition, one or more readertype flags can be bitwise-OR(|) in flags, then API will search the specify multi-Types Card int cardmode 0x00: Bank Card 0x00:M1(Mifare-One) Card 0x08: ID Card(Chinese Standard) 0x11: NFC Tag 0x40:sle4442/4428/at88sc102/AT24Cxx Apple VAS: 0x04:vas or payment 0x14:vas and payment 0x24:vas only 0x34:payment only

	String appName Package Name for the APK
Parameter Output	byte[] outData = new byte[256]; int[] length = new int[1];
Result	int result = outData[0];
Remarks	outData[0]: 0x00 Got Card Magnetic Track Data, Data Encrypt Success 0x01 Read Card failed 0x02 Got Card Magnetic Track Data, Data Encrypt failed 0x03 Timeout 0x04 Cancel Searching Card 0x05 Contact IC card insert 0x15 4442 Card detected. 0x25 4428 Card detected. 0x35 AT88SC102 Card detected. 0x45 AT24CXX Card detected. 0x07 Contactless IC Card detected 0x27 multi-Contactless IC Card are detected 0x37 M1-S50 Card detected. 0x47 M1-S70 Card detected. 0x57 UL(UltraLight) /NTAG203 Card detected. 0x87 DesFire Card detected.
Notes	Could use this API in a looper for Continued-Search Card, Could Control the situation when and where to break the Loop. Prerequisite: Must call the API **openCloseCardReader** before this
Furthermore	If you use the API, search out: ● A IC Card: If IC card reading is successful, the length of ATR and ATR data will be returned directly. see SDK - ii for card TAG information. ● A Magnetic Stripe Card: outData will include the Magnetic Track Data, you can call the API parseMagnetic to parse Magnetic Track Data ● A AT24CXX Card: Card type parser in appendix 1.

2.3.2 ReadCard (Blocking-Mode)

Function Name	**readCard**
Parameter Input	int cardtype; 0x00:Bank Card 0x01:Industry Card int cardMode; 0x0100:Contact IC Card 0x4100: PSAM Card 1 0x8100: PSAM Card 2

	0x0200: Contactless IC Card 0x0400: Magnetic stripe card 0x0200: M1(Mifare-One) Card 0x0202: Felica Card 0x0208: ID Card(Chinese Standard) 0x0211: NFC Tag 0x0140:sle4442/4428/at88sc102/AT24Cxx Apple VAS: 0x0204:vas or payment 0x0214:vas and payment 0x0224:vas only 0x0234:payment only In addition, one or more cardmode flags can be bitwise-OR(|) in flags, then API will search the specify multi-Types Card int timeOut TimeOut times(second) String appName Package Name for the APK
Parameter Output	byte[] outData = new byte[256]; int[] length = new int[1];
Result	int result; 0:Success Others: Not success
Notes	This method will wait for return, The API will return: ● Detected a Card ● Time's up for Searching Card No need to invoke openCloseCardReader, recommend to use this method.
Remarks	outData[0]: 0x00 Got Card Magnetic Track Data, Data Encrypt Success 0x01 Read Card failed 0x02 Got Card Magnetic Track Data, Data Encrypt failed 0x03 Timeout 0x04 Cancel Searching Card 0x05 Contact IC card insert 0x15 4442 Card detected. 0x25 4428 Card detected. 0x35 AT88SC102 Card detected. 0x45 AT24CXX Card detected. 0x07 Contactless IC Card detected 0x27 multi-Contactless IC Card are detected 0x37 M1-S50 Card detected. 0x47 M1-S70 Card detected. 0x57 UL(UltraLight) /NTAG203 Card detected. 0x87 DesFire Card detected.

2.4 Parse Magnetic

Function Name	**parseMagnetic**
Parameter Input	byte[] outData; The return date for ReadCard(In Magnetic stripe card Case) int length Data Length
Parameter Output	byte[] mag1 = new byte[128]; Track 1 Data int[] magLen1 = new int[1]; Track 1 Data Length byte[] mag2 = new byte[64]; Track 2 Data int[] magLen2 = new int[1]; Track 2 Data Length byte[] mag3 = new byte[128]; Track 3 Data int[] magLen3 = new int[1]; Track 3 Data Length
Result	int result; 0:Success Others: Not success
Notes	After Read Magnetic Stripe Card Success. You can use this API to Parse the Magnetic Track Data, If the API return OK, you can logcat the track data as shown in the Remarks.

Remarks	`if (result == 0) {`
	`Log.v(TAG, "m1 = " + new String(mag1).substring(0,magLen1[0]));`
	`Log.v(TAG, "m2 = " + new String(mag2).substring(0,magLen2[0]));`
	`Log.v(TAG, "m3 = " + new String(mag3).substring(0,magLen3[0]));`
	`}`

2.5 Send APDU

Function Name	**sendAPDU**
Parameter Input	int cardType 0x0100: Contact IC Card 0x0200: Contactless IC Card For Contact IC Card: If you want to choose the Slot where the APDU Sending to, you shall set the highest bit to 1, as shown here: 0x8100(1000,0001,0000,0000): sending APDU to the ICC 0x9100(1001,0001,0000,0000): sending APDU to the PSAM1 0xA100(1010,0001,0000,0000): sending APDU to the PSAM2 If the highest bit set to 0, it will use last slot you choose to send the APDU. byte[] inData Send Data int inLen Send Data Length
Parameter Output	byte[] outData = new byte[128]; Response Data int[] outLen = new int[1]; Return Data Length
Result	int result; 0:Success Others: Not success

Notes	You must open the target Slot before you sending the APDU. you can send the APDU to Contact IC Card/Contactless IC Card/PSAM Card 1/PSAM Card 2
Remarks	outData[0-1] 0x0000: Success 0x0001: APDU execution not success 0x006C: Variable-length data length error 0x0051: not standard card 0x005A: Not success (communication fail) 0x0005: Data Length error outData[2-3] Total Data Length for APDU Response

2.6 Get Contactless card SN/UID information

Function Name	**getCardSNFunction**
Parameter Input	N/A
Parameter Output	byte[] outData = new byte[16]; Response Data int[] outLen = new int[1]; Return Data Length
Result	int result; 0:Success Others: Not success
Notes	You shall call `openCloseCardReader` or `readCard` to active the card-slot before you call this API to get the SN/UID information.

2.7 PICC Detect

Function Name	**piccDetect**
Parameter Input	N/A
Parameter Output	N/A
Result	int result; 0x00: Can't detect Contactless Card 0x01: Contactless Card detected 0x27: Multi-cards detected

2.8 ICC Detect

Function Name	**iccDetect**
Parameter Input	N/A
Parameter Output	N/A
Result	int result; 0x00: Can't detect Contact Card 0x01: Contact Card detected
Notes	This API only deal with ICC slot, can't deal with PSAM card slot

2.9 ID Card Detect

Function Name	**idcDetect**
Parameter Input	N/A
Parameter Output	N/A
Result	int result; 0x00: Can't detect ID Card(Chinese Standard) 0x01: ID Card detected.

2.10 IC Card Power UP/Down

Function Name	**icsLotPower**
Parameter Input	int slotType CardType 0x01:Contact IC Card int operationType Operation Code 0x01:Power Down 0x02:Power UP int time time interval(Second)
Parameter Output	byte[] ATRData = new byte[128]; ATR Data int[] ATRLen = new int[1]; ATR Data Length
Result	int result; 0:Success Others: Not success

2.11 NFC/ MifareUL Write Block Data

Function Name	**NFCTagWriteBlock**
Parameter Input	int tagId Block Number byte[] inData Data need to Write In
Parameter Output	N/A
Result	int result; 0:Success Others: Not success

2.12 NFC/ MifareUL Read Block Data

Function Name	**NFCTagReadBlock**
Parameter Input	int tagId Block Number
Parameter Output	byte[] outData = new byte[128]; Response Data int[] outLen = new int[1]; Data Length
Result	int result; 0:Success Others: Not success

2.13 Felica Open

Function Name	**Felica_Open**
Parameter Input	int cardSlotNums Block Number int sysEncode System Encode int respType Response Format
Parameter Output	int[] outCardNums = new int[1]; Detected Card Numbers byte[] outData = new byte[128]; Response Data int[] outDataLen = new int[1]; Response Data Length
Result	int result; 0:Success Others:Not success

2.14 Felica Operation

Function Name	**Felica_Transmit**
Parameter Input	byte[] inData Operation Data int inDataLen Data Length
Parameter Output	byte[] outData = new byte[520]; Response Data int[] outDataLen = new int[1]; Data Length
Result	int result; 0:Success Others: Not success

2.15 Verify Logic Card Password

Function Name	**VerifyLogicCardPwd**
Parameter Input	byte[] pwd Password 3 Bytes User Password For 4428/AT88SC102 Card, the API only deal with the last 2 Bytes.
Parameter Output	N/A

Result	int result; 0x00:Success 0x81: Card can't supported this operation Others: Not success

2.16 Write Logic Card Data

Function Name	**WriteLogicCardData**
Parameter Input	byte[] pwd Password 3 Bytes User Password For 4428/AT88SC102 Card, the API only deal with the last 2 Bytes. For AT24CXX Card, set to 0x000000 (Because AT24CXX Card doesn't have user password) int addr Address 2 Bytes 4442 Card:0~255 4428 Card:0~1023 AT88SC102 Card:0~127 AT24CXX Card: Unrestricted Address int dataLen Data Length byte[] data Data
Parameter Output	N/A
Result	int result; 0:Success Others: Not success

2.17 Read Logic Card Data

Function Name	**readLogicCardData**
Parameter Input	int addr Address 2 Bytes 4442 Card:0~255 4428 Card:0~1023 AT88SC102 Card:0~127 AT24CXX Card: Unrestricted Address int len Expected return Data Length
Parameter Output	byte[] outData = new byte[64]; Response Data int[] outLen = new int[1]; Data Length
Result	int result; 0:Success Others: Not success

2.18 M1 Card Key Authentication

Function Name	**m1CardKeyAuth**
Parameter Input	int keyType KeyType: 0x41: Key A 0x42: Key B int blocknum Block Number int keyLen Length of the key byte[] key Key Data int snLen SN Data Length byte[] SN SN Data
Parameter Output	N/A
Result	int result; 0:Success Others: Not success

2.19 M1 Card Write Block Data

Function Name	**m1CardWriteBlockData**
Parameter Input	int blocknum block number int dataLen Data Length byte[] data Data Length must be 16
Parameter Output	N/A
Result	int result; 0:Success Others: Not success

2.20 M1 Card Read Block Data

Function Name	**m1CardReadBlockData**
Parameter Input	int blocknum block number
Parameter Output	int[] dataLen = new int[1]; Data Length byte[] data = new byte[20]; Data Length must be 16
Result	int result; 0:Success Others: Not success

2.21 M1 Card Value Operation

Function Name	**m1CardValueOperation**
Parameter Input	int operType Operation Type 0x2B: Add Value (Cash In) 0x2D: Sub Value(Purchas) int operBlocknum The block need to handle int operAmount Amount int operWritenum The block need to Write the Operation Result
Parameter Output	N/A
Result	int result; 0:Success Others: Not success

2.22 DesFire Select App

Function Name	**DesFire_SelApp**
Parameter Input	int aidLen length for the AID (Must be 3) byte[] aidData AID
Parameter Output	int[] outDataLen = new int[1]; Data Length byte[] outData= new byte[128]; Response Data
Result	int result; 0:Success Others: Not success

2.23 DesFire Authtication

Function Name	**DesFire_Auth**
Parameter Input	int keyNo Key Number Bit0~3 = 0x00~0x0D Bit4~7 = 0x00 Legacy = 0x01 ISO = 0x0A AES

	int keyType int keyLen byte[] keyData	Key Type(reserve, the default values is 0) Key Length (0x10/0x18/0x08) Key data
Parameter Output	N/A	
Result	int result;	0:Success Others: Not success

2.24 DesFire Get Card Info

Function Name	**DesFire_GetCardInfo**	
Parameter Input	int mode int id	Mode 0 GetVersion 1 GetApplicationIDs 2 GetKeyVersion 3 GetKeySettings 4 GetFileIDs 5 GetFileSettins If *mode* is 2 or 5, it shall set to KeyNo or FileID Other mode value, it shall set to 0
Parameter Output	int[] outDataLen = new int[1]; byte[] outData= new byte[128];	Data Length Response Data
Result	int result;	0:Success Others: Not success

2.25 DesFire Read File

函数名	**DesFire_ReadFile**	
入参	int fileType int fileId int offset int readLen	文件类型 0 Std/BackFile 1 RecordFile 2 ValueFile 文件 ID 起始位置偏移量 读取长度
出参	byte[] outData = new byte[512]; int[] outLen = new int[1];	返回数据 数据长度
结果	int result;	0:成功 非 0:失败

2.26 DesFire Write File

Function Name	**DesFire_WriteFile**	
Parameter Input	int fileType	File Type

	0 Std/BackFile 1 RecordFile; int fileId File ID int offset Offset int writeLen Write Length byte[] writeData Write Data
Parameter Output	byte[] outData = new byte[512]; Response Data int[] outLen = new int[1]; Data Length
Result	int result; 0:Success Others: Not success

2.27 DesFire Value File Operation

Function Name	**DesFire_ValueFileOpr**
Parameter Input	int fileType File Type 1 Credit 2 Debit 3 Limited Credit int fileId File ID byte[] operateValue Operation Value (Data Length is 4 Bytes)
Parameter Output	byte[] outData = new byte[512]; Response Data int[] outLen = new int[1]; Data Length
Result	int result; 0:Success Others: Not success

2.28 DesFire Confirm and Cancel

Function Name	**DesFire_Comfirm_Cancel**
Parameter Input	int fileType File Type 1: Confirm 2: Cancel
Parameter Output	byte[] outData = new byte[512]; Response Data int[] outLen = new int[1]; Data Length
Result	int result; 0:Success Others: Not success

2.29 Desfire ISO7816

Function Name	**DesFire_ISO7816**
Parameter Input	byte[] apdu APDU Command
Parameter Output	byte[] outData = new byte[64]; Response Data int[] outLen = new int[1]; Data Length
Result	int result; 0:Success Others :Not success

2.30 DesFire Delete File

Function Name	**DesFire_DelFile**
Parameter Input	int fileType File Type 0 Application 1 File 2 Format card byte[] aid_fidData AID (when the File Type is 1, Shall reserve the Last 2 Bytes)
Parameter Output	int[] outDataLen = new int[1]; Data Length byte[] outData= new byte[128]; Response Data
Result	int result; 0:Success Others: Not success

2.31 Remaining number of times that logic contact card read password

Function Name	**Logic_ReadPWDegree**
Parameter Input	N/A
Parameter Output	byte[] outData= new byte[64]; Response Data
Result	int result; 0:Success Others: Not success

2.32 Modify the password of the Logical contact card

Function Name	**Logic_ModifyPW**
Parameter Input	byte[] inData = new byte[64]; inData int[] inLen = new int[1]; dataLen
Parameter Output	N/A
Result	int result; 0:Success Others: Not success

2.33 Logical Encryption Card Operation(communication base on I2C)

Function Name	**Logic_I2C**
Parameter Input	int type operation time sequence: 0 Start 1 output 1 byte data and receive response 2 input 1 byte data and send Ack 3 input 1 byte data without send Ack 4 quit byte[] inData OutputData(Exists when the time sequenceis 1)
Parameter Output	byte[] outData = new byte[64]; Response Data Time sequence:0x00 N/A Time sequence:0x01 0x00—FALSE 0x01—TRUE Time sequence:0x02/0x03 InputData Time sequence:0x04 N/A
Result	int result; 0:Success

	Others:Not Success

3. Key Injection

Note:

You shall new a Class Key Instance, before calling the Key Injection API.
And the instantiation shall be process in a Thread routine,
as shown the code segment below:

```
new Thread(new Runnable() {
    @Override
    public void run() {
        mKey = new Key(getApplicationContext());
    }
}).start();
```

3.1 Erase PED

Function Name	**erasePED**
Parameter Input	N/A
Parameter Output	N/A
Result	int result; 0:Success Others: Not success
Notes	Erase PED will clear out all key Data

3.2 Import IPEK(KEK Encryption)

Function Name	**InjectIPEKByKEK**
Parameter Input	int AlgorithmType Algorithm type（Only support 3DES） 0x01:DES 0x02:3DES 0x04: AES 0x10:SM4 Int IPEKLength ipek data length （for 3des value is 8） byte[] IPEKData ipek ciphertext int InputChecksumLength check value length byte[] InputChecksum check value data（8 bytes 0x00 by ipek encryption） String packageName package name,The same as Kek package name
Parameter Output	N/A
Result	int result; 0: Success Others:Not Success

3.3 Import/Update Symmetric Encryption Algorithm Key

Function Name	updateKeyEx		
Parameter Input	int	keyType	Key Type 0x01:TLK 0x03:MK/SK TMK 0x04:DUKPT IPEK 0x05:TR-31 KBPK 0x08:PEK(PIN Key) 0x09:DEK(TEK) 0x19:DDEK(TDK) 0x0A:MAK(MAC Key) 0x1A:DMAK
	int	encryptKeyType	Key Protect Type 0x00:IN Plain Text(useless for working key) 0x01:TLK 0x03:TMK
	byte[]	certData	reserve, new byte[8]
	byte[]	key	Key Values
	boolean	isCheck	KCV checking option: true: Do verify false: Do Not Verify
	int	checkvallen	KCV length
	byte[]	checkval	KCV Value
	String	packageName	Package name for the APP
	int	specifyId	Key Index (If set to 0, SDK will specify a random and unique number)
Parameter Output	N/A		
Result	int	result;	0:Success Others: Not success
Notes	The API will import a Key, and save in the kernel. If the key is exist with same package name, the API will overwrite the key. The key will default store as a 3DES key.		

3.4 Update Key With Algorithm

Function Name	updateKeyWithAlgorithm		
Parameter Input	int	keyType	Key Type 0x01:TLK 0x03:MK/SK TMK 0x04:DUKPT IPEK 0x05:TR-31 KBPK 0x08:PEK(PIN Key)

	0x09:DEK(TEK) 0x19:DDEK(TDK) 0x0A:MAK(MAC Key) 0x1A:DMAK int encryptKeyType Key Protect Type 0x00:IN Plain Text(useless for working key) 0x01:TLK 0x03:TMK byte[] certData reserve, new byte[8] byte[] key Key Values boolean isCheck KCV checking option: true: Do Verify false: Do Not Verify int checkvallen KCV length byte[] checkval KCV Value String packageName Package name for the APP int specifyId Key Index (If set to 0, SDK will specify a random and unique number)
Parameter Output	N/A
Result	int result; 0:Success Others:Not success
Notes	The API will import a Key, and save in the kernel. If the key is exist with same package name, the API will overwrite the key.

3.5 Check Key Exist

Function Name	**checkKeyExist**
Parameter Input	String packageName Package name for the APP int keyType Key Type 0x01:TLK 0x03:MK/SK TMK 0x04:DUKPT IPEK 0x05:TR-31 KBPK 0x08:PEK(PIN Key) 0x09:DEK(TEK) 0x19:DDEK(TDK) 0x0A:MAK(MAC Key) 0x1A:DMAK
Parameter Output	N/A
Result	int result; 0: Success 1: the function parameter isn't correct

	2: kernel can't find the key valuse -1: SDK can't find the key in the specify package name

3. Get Key KCV

Function Name	getKeyKCV
Parameter Input	String packageName Package name for the APP int keyType Key Type 0x01:TLK 0x03:MK/SK TMK 0x04:DUKPT IPEK 0x05:TR-31 KBPK 0x08:PEK(PIN Key) 0x09:DEK(TEK) 0x19:DDEK(TDK) 0x0A:MAK(MAC Key) 0x1A:DMAK int specifyId same to inject key Example: Arithmetic: 3des keyType: 0x03:MK/SK TMK KCV of TMK:c8f7c5a8f7712bb0b6 TLK(plaintext):11111111111111111111111111111111 TMK(ciphertext):87432C07DA6BC82DCB48C1168061F6FE
Parameter Output	byte[] outData = new byte[64]; int[] outLen = new int[1];
Result	int result; 0:Success Others: Not success

4. DUKPT

4.1 Inject IKSN

Function Name	InjectIKSN
Parameter Input	String packageName Package name for the APP int IKSNLen IKSNData Length byte[] IKSNData IKSN Data
Parameter Output	N/A
Result	int result; 0:Success Others: Not success

4.2 Get KSN

Function Name	GetKSN
Parameter Input	String packageName Package name for the APP
Parameter Output	int[] IKSNLen = new int[1]; Parameter IKSNData Length

	byte[] IKSNData = new byte[16]; 2 bytes for IKSN Length + IKSN Data
	
	Example:
	Step 1, Inject KSN
	String azKSN = "629949012c0000000001";
	byte[] IKSNData = ByteUtil.hexString2ByteArray(azKSN);
	Int IKSNLength = IKSNData.length;
	try {
	ret = mKey.InjectIKSN(packageName, IKSNLength, IKSNData);
	} catch (RemoteException e) {
	e.printStackTrace();
	}
	Step 2, Get KSN
	byte[] IKSNData = new byte[32];
	int[] IKSNLen = new int[1];
	try {
	mKey.GetKSN(packageName, IKSNData, IKSNLen);
	} catch (RemoteException e) {
	e.printStackTrace();
	}
	You will get : IKSNData =[0x00 0x0a 0x62 0x99 0x49 0x01 0x2c 0x00 0x00 0x00 0x00 0x01]. The ” 0x00 0x0a” is length of KSN
	IKSNLen[0] = 12; is IKSNData.length

Result	int result; 0:Success Others: Not success

4.3 Increase KSN

Function Name	**IncreaseKSN**
Parameter Input	String packageName Package name for the APP
Parameter Output	N/A
Result	int result; 0:Success Others :Not success

5. TR-31

5.1 Inject Key Block

Function Name	**injectKeyBlock**
Parameter Input	String packageName Package name for the APP int KeyBlockLength Key Block Length byte[] KeyBlockData Key Block Data
Parameter Output	N/A
Result	int result; 0:Success Others: Not success

6. PINPAD

app call startPinInput

1)sp will generate a random button sequence
2)sp return 0b02 01 call back to tell app the button sequence
3)app will use the button sequences to change the display value of the button
4)app use generatePinPreparedData to get button coordinate
5)app return the button coordinate to sp
6)then sp wait for cardholder button click
7)sp tell app pin input update
8)sp tell app pin input finish

6.1 Setup the PINPAD layout

Function Name	generatePINPrepareData
Parameter Input	Button btnb1 Button 1 Button btnb2 ... Button btnb9 Button btnb0 Button btnCancel Button Cancel Button btnConfirm Button Confirm Button btnClean Button Clear Button btnBack Button Backspace Activity mActivity Activity Class Instance Reference
Parameter Output	N/A
Result	N/A
Notes	The API notify the layout coordinate to kernel. PIN input step will handle by kernel. Before kernel return, do not change the PINPAD layout, and do not do screen rotation.

6.2 Invoke PINPAD(Start PIN input)

6.2.1 Key Type: MK/SK

Function Name	startPinInput		
Parameter Input	int	timeOut(S)	Timeout Times(Second)
	String	packageName	Package name for the APP (Shall same to the PEK)
	int	byPass	Bit 0: whether bypass mode is supported or not, 1 is supported; Bit 2: setting this bit means that only the parameters of pinpad are set (which takes effect when the kernel calls it), and it is not a real call. Bit 4:: whether pin full automatic confirmation is supported, 1 is supported; Bit 5: the password keyboard is not rel-layout(i.e. | 0x20); Bit6:set to 1 means set the pinpad with buzzing Bit 7: 0:clear button same as cancel 1:clear button is still clear button (i.e. |0x80)
	int	pinLenMin	Min Length for the PIN (Not less than 4)
	int	pinLenMax	Max Length for the PIN (Max Value is 12)
	int	blockFormat	PinBlock Format: 0x00 ISO9564 Format 0 0x01 ISO9564 Format 1 0x03 ISO9564 Format 3
	int	forMatData	reserve, new byte[8]
	int	panLen	Length for PAN
	byte[]	panData	PAN Data
	ICallbackListener callback		PIN input callback Reference; Refer to the Appendix 2;
Parameter Output	N/A		
Result	The PIN Input Result will return by the Callback.		

6.2.2 Offline PIN

Function Name	startPinInputOff		
Parameter Input	int	timeOut(S)	Timeout Times(Second)
	int	byPass	Bit 4:: whether pin full automatic confirmation is supported, 1 is

	supported; Bit 5: the password keyboard is not rel-layout(i.e. | 0x20); Bit6:set to 1 means set the pinpad with buzzing int pinLenMin Min Length for the PIN (Not less than 4) int pinLenMax Max Length for the PIN (Max Value is 12) int exponet Public Key Exponet byte[] publicKey Public Key ICallbackListener callback PIN input callback Reference; Refer to the Appendix 2;
Parameter Output	N/A
Result	The PIN Input Result will return by the Callback.

6.2.3 Offline PIN(Single)

Function Name	**startPinInputOffLineForSingle**
Parameter Input	int timeOut(S) Timeout Times(Second) int byPass Bit 4:: whether pin full automatic confirmation is supported, 1 is supported; Bit 5: the password keyboard is not rel-layout(i.e. | 0x20); Bit6:set to 1 means set the pinpad with buzzing Int pinLenMin Min Length for the PIN (Not less than 4) Int pinLenMax Max Length for the PIN (Max Value is 12) Int exponet Public Key Exponet byte[] publicKey Public Key int pinBlock PinBlock Mode outData IC Card Data outDataLen IC Card Data Length ICallbackListener callback PIN input callback Reference; Refer to the Appendix 2;
Parameter Output	N/A
Result	The PIN Input Result will return by the Callback.

6.2.4 Get Offline PIN retry Counter

Function Name	**getOfflinePINTryCounter**
Parameter Input	int reserved reserved
Parameter Output	byte[] outData Response Data： byte 1-2 SW1、SW2、May be empty when executing an error, byte 3

	is PIN retry count		
	int[]	outDataLen	Response Data Length
Result	int	result;	0: Success Others:Not Success

6.2.5 Key Type: DUKPT

Function Name	startPinInputForIPEK		
Parameter Input	int	timeOut(S)	Timeout Times(Second)
	String	packageName	Package name for the APP (Shall be same to the PEK)
	int	byPass	Bit 0: whether bypass mode is supported or not, 1 is supported; Bit 2: setting this bit means that only the parameters of pinpad are set (which takes effect when the kernel calls it), and it is not a real call. Bit 4:: whether pin full automatic confirmation is supported, 1 is supported; Bit 5: the password keyboard is not rel-layout (i.e. | 0x20); Bit6:set to 1 means set the pinpad with buzzing Bit 7: 0:clear button same as cancel 1:clear button is still clear button (i.e. |0x80)
	int	pinLenMin	Min Length for the PIN(Not less than 4)
	int	pinLenMax	Max Length for the PIN(Max Value is 12)
	int	blockFormat	PinBlock Format: 0x00 ISO9564 Format 0 0x01 ISO9564 Format 1 0x03 ISO9564 Format 3
	int	forMatData	reserve, new byte[8]
	int	panLen	Length for PAN
	byte[]	panData	PAN Data
	ICallbackListener callback		PIN input callback Reference; Refer to the Appendix 2;
Parameter Output	N/A		
Result	The PIN Input Result will return by the Callback.		

7. Data Encryption/Decryption

7.1 MK/SK Data Encryption/Decryption

Function Name	**dataEnDecryptEx**
Parameter Input	int algorithmFlag Algorithm Type 0x01:DES 0x02:3DES 0x04:AES 0x10:SM4 int operationMode Operation Mode 0 :Encrypt 1:Decrypt String pkgeName Package name for the APP (Shall be same to the DEK/DDEK) int encryptMode Encrypt Mode 1:ECB 2:CBC int vectorLen set to 8 int vectorData reserve, new byte[8] int dataLen Data Length byte[] data Data need to process int pddMode Padding Mode 0: MODE_NONE, N/A No Padding; The data length needs to equal to times of 8 bytes. 1:MODE_9797, Padding compatible with ISO9797 This means in practice that the first byte is a mandatory byte valued '80' (Hexadecimal) followed, if needed, by 0 to N-1 bytes set to '00', until the end of the block is reached. The block size needs to be times of 8. In the following example the block size is 8 bytes and padding is required for 4 bytes ... | DD DD DD DD DD DD DD DD | DD DD DD DD 80 00 00 00 | The next example shows a padding of just one byte ... | DD DD DD DD DD DD DD DD | DD DD DD DD DD DD DD 80 | The Last example shows a padding required for 8 bytes ... | DD DD DD DD DD DD DD DD | 80 00 00 00 00 00 00 00 | 2:MODE_2, Prefix 2 Bytes Data Length, others are same rule as MODE_9797
Parameter Output	byte[] outdata = new byte[512]; Response Data int outLen = new int[1]; Data Length

Result	int result; 0:Success Others: Not success

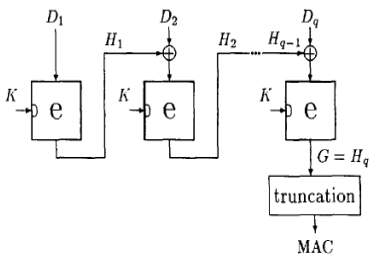
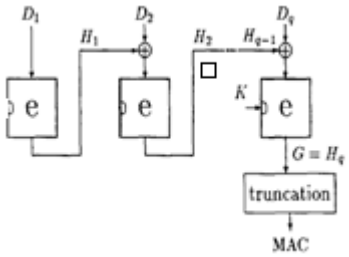
7.2 DUKPT Data Encryption/Decryption for IPEK

Function Name	**dataEnDecryptForIPEK**		
Parameter Input	int	algorithmFlag	Algorithm Type 0x01:DES 0x02:3DES 0x04:AES 0x10:SM4
	int	operationMode	Operation Mode 0: Encrypt 1: Decrypt
	String	pkgeName	Package name for the APP (Shall be same to the DEK/DDEK)
	int	encryptMode	Encrypt Mode 1:ECB 2:CBC
	int	vectorLen	set to 8
	int	vectorData	reserve, new byte[8]
	int	dataLen	Data Length
	byte[]	data	Data need to process
	int	pddMode	Padding Mode 0: MODE_NONE, N/A No Padding; The data length needs to equal to times of 8 bytes. 1:MODE_9797, Padding compatible with ISO9797 This means in practice that the first byte is a mandatory byte valued '80' (Hexadecimal) followed, if needed, by 0 to N-1 bytes set to '00', until the end of the block is reached. The block size needs to be times of 8. In the following example the block size is 8 bytes and padding is required for 4 bytes ... | DD DD DD DD DD DD DD DD | DD DD DD DD 80 00 00 00 | The next example shows a padding of just one byte ... | DD DD DD DD DD DD DD DD | DD DD DD DD DD DD DD 80 | The Last example shows a padding required for 8 bytes ... | DD DD DD DD DD DD DD DD | 80 00 00 00 00 00 00 00 |

	2:MODE_2, Prefix 2 Bytes Data Length, others are same rule as MODE_9797
Parameter Output	byte[] outdata = new byte[512]; Response Data int outLen = new int[1]; Data Length
Result	int result; 0:Success Others: Not success

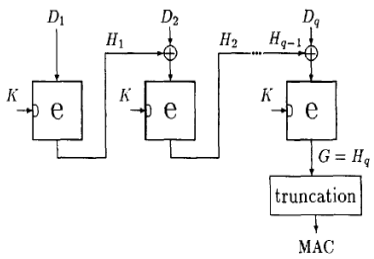
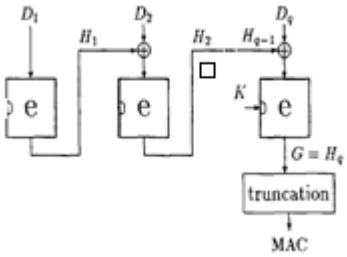
8. Get MAC

8.1 MK/SK: Get MAC with Algorithm

Function Name	**getMacWithAlgorithm**		
Parameter Input	String	pkgeName	Package name for the APP (Shall be same to the MAK)
	int	algorithm	Algorithm Type 0x01: DES 0x02:3DES 0x04: AES 0x10: SM4
	int	vectorLen	Set to 8 now
	int	vectorData	reserve, new byte[8]
	int	dataLen	Data Length
	byte[]	data	Data need to process
	int	macMode	MAC Mode 0x00: ISO/IEC 9797 MAC Algorithm 1  Figure 2 — MAC Algorithm 1. 0x01:  0x02: ANSIX9.19 0x04: China Union Pay MAC (101 Mode) (In DUKPT case 0x04: Request 0x14: Response)	

Parameter Output	byte[]	outdata	= new byte[128];	MAC Value
	int	outLen	= new int[1];	Data Length
Result	int	result;	0:Success	Others: Not success

8.2 DUKPT: Get MAC for IPEK

Function Name	getMacForIPEK		
Parameter Input	String	pkgeName	Package name for the APP (Shall be same to the MAK)
	int	algorithm	Algorithm Type 0x01: DES 0x02: 3DES 0x04: AES 0x10: SM4
	int	vectorLen	Set to 8 now
	int	vectorData	reserve, new byte[8]
	int	dataLen	Data Length
	byte[]	data	Data need to process
	int	macMode	MAC Mode 0x00: ISO/IEC 9797 MAC Algorithm 1 Request Or 0x10: ISO/IEC 9797 MAC Algorithm 1 Response  Figure 2 — MAC Algorithm 1. 0x01: Request Or 0x11: Response  0x02: ANSIX9.19 Standard Request Or 0x12: ANSIX9.19 Standard Response 0x04: China Union Pay MAC (101 Mode) Request

	Or 0x14: China Union Pay MAC (101 Mode) Response
Parameter Output	byte[] outdata = new byte[128]; MAC Value int outLen = new int[1]; Data Length
Result	int result; 0:Success Others: Not success

9. Printer

Note:

Printer Class include the printer related API, support the Multi-Language print.
You shall new a Class Printer Instance, before calling the Printer API.
And the instantiation shall be process in a Thread routine,
as shown the code segment below:

```
new Thread(){
    @Override
    public void run() {
        mPrinter = new Printer(getApplicationContext());
    }
}.start();
```

The general printing invoke flow:

1. Get the Printer Status
2. Printer Initialization
3. Clear Print Data Cache
4. Load Print Data (Text/Picture/Barcode/QR code)
5. Printer print and Roll-Out Paper
6. Finish Printing

9.1 Get Printer Status

Function Name	**getPrinterStatus**
Parameter Input	N/A
Parameter Output	byte[] status = new byte[1]; Response Data: 0x00: Printer status OK 0x01: Parameter Error 0x06: Not available 0x8A: Out of Paper 0x8B: Overheat
Result	int result = status[0];

9.2 Printer Initialization

Function Name	**printInit**
Parameter Input	N/A
Parameter Output	N/A
Result	int result; 0:Success Others: Not success

9.3 Set GrayLevel

Function Name	**setGrayLevel**
Parameter Input	0:Gets the current grayscale level 1~3:grayscale level 1:high 3:low
Parameter Output	N/A
Result	int result; when input parameter is 0:return the current grayscale level When input parameter is 1-3:return result of execution 0:Success Others: Not success

9.4 Clear Print Data Cache

Function Name	**clearPrintDataCache**
Parameter Input	N/A
Parameter Output	N/A
Result	int result; 0:Success Others: Not success

9.5 Load Print Data

9.5.1 Print String

1. Normal Text

Function Name	**printString**
Parameter Input	String content; Text Data int fontSize Font Size Align align Align boolean isBold Bold Font boolean isItalic Italic Font
Parameter Output	N/A
Result	int result; 0:Success Others: Not success

2. Text with Format

Function Name	**printStringExt**
Parameter Input	String content; Text Data int leftOffset Offset from left boundary float letterWidth Width for the letter space float lineSpacing Height of row

	Font	font	Font Style DEFAULT: Default Font MONOSPACE: Monospaced font SERIF: SERIF font SANS_SERIF: SANS SERIF font DEFAULT_BOLD: BOLD font
	int	fontSize	font Size
	Align	align	align
	boolean	isBold	Bold Font option
	boolean	isItalic	Italic Font option
	boolean	isUnderLine	UnderLine option
Parameter Output	N/A		
Result	int	result;	0:Success Others: Not success

3. **Print Both side**

Function Name	**print2StringInLine**		
Parameter Input	String	content1;	Left Size Text
	String	content2;	Right Size Text
	float	lineSpacing	height of row
	Font	font	Font Style DEFAULT: Normal Font MONOSPACE: monospaced font SERIF: SERIF font SANS_SERIF: SANS SERIF font DEFAULT_BOLD: BOLD font
	int	fontSize	font Size
	Align	align	align
	boolean	isBold	Bold Font option
	boolean	isItalic	Italic Font option
	boolean	isUnderLine	UnderLine option
Parameter Output	N/A		
Result	int	result;	0:Success Others: Not success

4. **Separate multiple columns**

Function Name	**printMultiseriateString**		
Parameter Input	Int[]	proportionArray;	A text scale array that corresponds to a text index
	String[]	contentArray;	A text array that corresponds to a proportional index
	int	fontSize	content size

	Align align display position boolean isBold bold or not boolean isItalic Italic or not
Parameter Output	N/A
Result	int result; 0:Success Others: Not success

9.5.2 Print Image

Function Name	**printImageBase**
Parameter Input	Bitmap bitmap Class Bitmap Instance (prefer .png format) int width width of the bitmap int height height of the bitmap Align align align int offset offset
Parameter Output	N/A
Result	int result; 0:Success Others: Not success

Remarks	try {
	InputStream inputStream = getAssets().open("logo.png");
	Bitmap bitmap = BitmapFactory.decodeStream(inputStream);
	mPrinter.printImageBase(bitmap, 100, 100, Align.LEFT, 0);
	bitmap.recycle();
	}catch (IOException e){
	e.printStackTrace();
	}catch (RemoteException ex){
	ex.printStackTrace();
	}

9.5.3 Print Barcode

Function Name	**printBarCodeBase**
Parameter Input	String content The Text Context for Barcode BarcodeType type Barcode Type BarcodeWidth width width int height height int offset offset
Parameter Output	N/A
Result	int result; 0:Success Others: Not success
Notes	Not support PDF417

9.5.4 Print PDF417 Code(Fixed width and height)

Function Name	**printPDF417**
Parameter Input	String content The text context for PDF417

Parameter Output	N/A

9.5.5 Print PDF417 Code

Function Name	printPDF417
Parameter Input	String content The text context for PDF417 int width the width of PDF417 int height the heighth of PDF417（Invalid）
Parameter Output	N/A

9.5.6 Print QR Code

Function Name	printQRCode
Parameter Input	String content The Text Context for QR Code int width width of square Align align align
Parameter Output	N/A
Result	int result; 0:Success Others: Not success

Function Name	printQRCode
Parameter Input	String content The Text Context for QR Code int width width of square Align align align boolean isPadding QRCode Whether there is a margin
Parameter Output	N/A
Result	int result; 0:Success Others: Not success

9.6 Print Paper (Print and Roll-Out Paper)

9.6.1 Print Paper (Normal Mode Print)

Function Name	printPaper
Parameter Input	int step Row number of margins
Parameter Output	N/A
Result	int result; 0:Success Others: Not success
Notes	Invoke this API to Print and roll-out paper after loading print data.

9.6.2 Print Paper (Roll-Out Paper and Check Card)

Function Name	printPaper_trade

Parameter Input	int trandType IC Card Type: 5.Contact IC Card 7.Contactless IC Card int step Row number of margins
Parameter Output	N/A
Result	int result; 0:Success Others: Not success
Notes	When a IC Card transaction Success, then start loading print Data, Invoke this API to Print and roll-out paper, The API also start checking the state for the specify IC Card Slot, and broadcast the state (The Checking premises is the Slot is open)

Remarks	Register the broadcast:
	mIntentFilter = new IntentFilter("com.wpos.printer_card");//Action
	registerReceiver(mReceiver, mIntentFilter);
	Instance a BroadcastReceiver:
	private BroadcastReceiver mReceiver = new BroadcastReceiver() {
	@Override
	public void onReceive(Context context, Intent intent) {
	if (intent != null) {
	//Broadcast loop check card status, 1: exist; 2: no
	int flag = intent.getIntExtra("printer_c",-1);
	if (flag == 1) {
	//"card into";
	}else if (flag == 2) {
	//"card out";
	}}}};

9.7 Print Finish

Function Name	**printFinish**
Parameter Input	N/A
Parameter Output	N/A
Result	int result; 0:Success Others: Not success
Notes	Finish the Print routine after print paper

9.8 Get/Clear Printer Mileage

Function Name	**Get_ClearPrinterMileage**
Parameter Input	int status State 0x00: Reset 0x01: Return Mileage
Parameter Output	byte[] outData = new byte[4]; Response Data (mm, millimeter)
Result	int result; 0:Success Others: Not success

Remarks	Return the Mileage (when status = 0x01) Log.d(TAG,"Printer Mileage: "+ByteUtil.*bytes2Int*(outData)+" mm");

9.9 Set Print Type

Function Name	setPrintType
Parameter Input	int type 0: Internal Printer 1:BlueTooth Printer 2:USB Printer
Parameter Output	N/A
Result	int result; 0:Success Others: Not success

9.10 Set Print Paper Type

Function Name	setPrintPaperType
Parameter Input	int type 0: 58mm paper 1:80mm paper
Parameter Output	N/A
Result	int result; 0:Success Others: Not success

9.11 Set Print Line Spacing

Function Name	setPrintLineSpacing
Parameter Input	Int type 0 no Spacing 1 low Spacint 2 high Spacint
Parameter Output	N/A
Result	int result; 0:Success Others: Not success

10. BlueTooth-Printer

The general Bluetooth printing invoke flow:

bluetooth printing initialization - clean up the print cache - content (text / picture) load - start bluetooth printing - finish bluetooth printing

10.1 Init Bluetooth Printer

Function Name	initBlueToothPrint
Parameter Input	ICommonCallback callback print callback: 0: OK 1: connection error 2: out of paper 3: off-line 4: print object not connected 10: print finish
Parameter Output	N/A
Result	int result; 0: success 1: Bluetooth is not open 2: No connected devices 3: connection failed 5: exception occurs 10:init repetition

10.2 Clear Print Data Cache/Load Print Data

The method is the same as the built-in printer

10.3 BlueToothprintStart

Function Name	`startBlueToothPrint`
Parameter Input	N/A
Parameter Output	N/A
Result	int result; 0:Success 5: Exception occurs 6: in printing 7: The device is not initialized 8: Connection exception, please reconnect.
Notice	Invoke this API to Print and roll-out paper after loading print data. The result will notify the app by CallBack

10.4 BlueToothprintFinish

Function Name	`finishBlueToothPrint`
Parameter Input	N/A
Parameter Output	N/A
Result	int result; 0:Success Others: Not success
Notes	Finish the Print routine after print paper

11. Scan Code

11.1 Init

Function Name	`decode_Init`
Parameter Input	IDecodeCallback callback
Parameter Output	N/A
Result	int result; 0:Success Others:Not success
Notes	Init the scan module

11.2 Close Scan module

Function Name	decode_Close
Parameter Input	N/A
Parameter Output	N/A
Result	int result; 0:Success others:Failed
Notes	Close Scanner, disconnect API

11.3 Scan Code(One Time)

Function Name	decode_StartSingleScan
Parameter Input	Int timeout Scan code time out(ms) maximum10*60*1000
Parameter Output	N/A
Result	int result; 0:Success others:Failed
Notes	Will Scan one time The result will return by the IDecodeCallback resultCallback

11.4 Scan Code(Multi-time) - only for mini2

Function Name	decode_StartSingleScan
Parameter Input	Int timeout Scan code time out(ms) maximum10*60*1000
Parameter Output	N/A
Result	int result; 0:Success others:Failed
Notes	The results of the initial multiple sweeps are returned in the IDecodeCallback resultCallback,The number of scan codes can be set

11.5 Loop Scan Code - only for mini2

Function Name	decode_StartContinuousScan
Parameter Input	Int timeout Scan code time out(ms) maximum10*60*1000
Parameter Output	N/A
Result	int result; 0:Success others:Failed
Notes	Start continuous many code ,The results are returned in the IDecodeCallback resultCallback,Need call decode_StopScan to stop .

11.6 Stop Scan Code

Function Name	decode_StopScan
Parameter Input	N/A
Parameter Output	N/A
Result	int result; 0:Success Others:Not Success
Notes	Cancel current operation. Can be used for single sweep, multiple sweep, continuous single sweep

11.7 Set Maximum Read Count of Multi-Time Scan - only for mini2 – test

Function Name	decode_SetMaxMultiReadCount
Parameter Input	Int max count
Parameter Output	N/A
Result	int result; 0:Success Others:Not Success
Notes	Set Maximum Times of Multi-Time Scan

11.8 Set lighting mode - only for mini2 - test

Function Name	decode_SetLightsMode
Parameter Input	Int mode 0 :Close the red sight and lighting 1 :Only the red sight 2 :only the lighting 3 :The red sight altermates with the lighting 4 :The red sight and lighting are present at the same time
Parameter Output	N/A
Result	int result;
Notes	Set the maximum number of multiple sweeps No maximum limit yet

11.9.Return values for Scan code module

Code int	description
0	Operation was successful
1	An image was requested using an invalid image region
2	Error detected in image engine driver
3	Image engine driver reported busy
4	Memory allocation failed
5	Image engine unable to decode a symbology
6	No image available
7	Could not communicate with imager
8	Not connected to image engine

9	One of the function parameters was invalid
10	The operation was not supported by the engine
11	Trigger state is false The user stops scanning the code
12	Trigger state is false
13	Requested IQ image too large
14	IQ image fail
15	Invalid structure size
16	Could not create async decode thread
17	Asynchronous decode was canceled
18	An exception was detected in the deoder
19	Scanned barcode is not a valid IQ host barcode
20	Error loading EXM file.
21	Not a valid configuration file.
22	Section missing from exm file.
23	Section missing from exm file.
24	Section missing from exm file.
101	Function call success
102	Scan code module opening
103	Parameter error
110	RemoteException
-1	Initialize
-2	Scanning code is not currently enabled
-10	Invalid device

12. Cortex Scan Code

Note:

Cortex Scan Code need import cortexdecodersdk-release.aar file

12.1 Init

Function Name	ScanDecoderLibrary.initDecoder
Parameter Input	Context context
Parameter Output	N/A
Result	ScanDecoderLibrary result;
Notes	Init the scan module,return ScanDecoderLibrary object

12.2 Scan Code

Function Name	ScanDecoderLibrary.scanCode
Parameter Input	IScanner.IScannerCallBack callback;
Parameter Output	N/A
Result	N/A

Notes	The result will return by the IScanner.IScannerCallBack

13. RSA

13.1 Import RSA Public Key

Function Name	ImportRSAPubKey
Parameter Input	int indexProcessMode Key Index Process Mode 1: Specify the index, if the index already exists, it is mandatory to cover 2: Specify the index, if the index already exists, do not force cover, return an error 3: Do not specify an index, auto generate and return the index 4: Update the specified index int keyIndex Public Key Index Under certain circumstances, this domain is invalid. The current key index should be 1~4, that is, only four keys can be stored int keyLen key length High byte first byte[] KeyModulus Key Modulus Length: Key digits/8 int KeyPubExp key Exponen High byte first
Parameter Output	byte[] outKeyIndex
Result	int result; 0:Success Others:Not Success
Notes	outKeyIndex[0] 00: Success, Others:Error code outKeyIndex[1] Return the assigned key index if the key index does not need to be returned, the value is meaningless

13.2 Erase RSA Key

Function Name	EraseAsymKey
Parameter Input	int KeyType Erase Key Type 1:RSA Key pair space 2:RSA Pubilc Key Space
Parameter Output	N/A
Result	int result; 0:Success Others:Not Success
Notes	Erase

13.3 Generate RSA Key Pair and return Public Key

Function Name	generateRSAKeyTest
Parameter Input	int indexProcessMode Key Index Process Mode 1: Specify the index, if the index already exists, it is mandatory to cover 2: Specify the index, if the index already exists, do not force cover, return an error 3: Do not specify an index, auto generate and return the index 4: Update the specified index int keyIndex Public Key Index Under certain circumstances, this domain is invalid. The current key index should be 1~4, that is, only four keys can be stored int keyLen key length High byte first
Parameter Output	byte[] outData return data int[] outDataLen return data Length
Result	int result; 0:Success Others:Not Success
Notes	return data resolve byte 1 key index byte 4 key Exponen byte N Key Modulus Length: Key digits/8

13.4 Perform RSA operations

Function Name	perfromRSAOperation
Parameter Input	int KeyType 0: Use a preset key pair, the key index is invalid 1: Use Import RSA Public Key 2: Use Generate RSA Key Pair int KeyIndex Key Index int arithmeticType arithmetic Type 1: Public Key 2: Private Key int fillMode fill Mode 1: RAW (No Fill) 2: PKCS#1 V1.5 int inDataLen input Data length byte[] inData input Data
Parameter Output	byte[] outData out Data int[] outDataLen out Data Length
Result	int result; 0:Success Others:Not Success
Notes	

Appendix

1.Remark for AT24Cxx Card

```
if (outData[0] == 0x45 && outData[1] == 0x01) {
    switch (outData[2]){
        case 0x01:
            //"AT24C01";
            break;
        case 0x02:
            //"AT24C02";
            break;
        case 0x03:
            //"AT24C04";
            break;
        case 0x04:
            //"AT24C08";
            break;
        case 0x05:
            //"AT24C16";
            break;
        case 0x06:
            //"AT24C32";
            break;
        case 0x07:
```

```
                //"AT24C64";
                break;
            case 0x08:
                //"AT24C128";
                break;
            case 0x09:
                //"AT24C256";
                break;
            case 0x0a:
                //"AT24C512";
                break;
}}
```

2.Remarks for PINPAD callback

Can use Handler to send Message to Main Thread to update UI, can get more detail in **wPos_SDKDemo project**

```
private ICallbackListener callback = new ICallbackListener.Stub() {
    @Override
    public int emvCoreCallback(int command, byte[] data, byte[] result, int[] resultlen) throws
RemoteException {
        if (command != mCore.CALLBACK_PIN)
            return -1;
        if (data[0] == mCore.PIN_CMD_PREPARE) {
            // Setup the PINPAD layout
            try {
                mCore.generatePINPrepareData(result, btnb1, btnb2, btnb3, btnb4,
         btnb5,btnb6, btnb7, btnb8, btnb9, btnb0, btncancel,
                  btnconfirm, btnclean,btnback,mContext);
                resultlen[0] = 105;
            } catch (Exception e) {
                Log.e("PINPad", "err " + e.toString());
            }
        } else if (data[0] == mCore.PIN_CMD_UPDATE) {
            // Update the Button
            result[0] = 0;
            resultlen[0] = 1;
        } else if (data[0] == mCore.PIN_CMD_QUIT) {
            // return the PINPAD latest state: Conform, Cancel, Timeout, Error, Others
              result[0] = 0;
            resultlen[0] = 1;
            if(data[1]==0){
            // Return the PinResutl
            String pin = BytesUtil.bytes2HexString
                        (Arrays.copyOfRange(data,4,4+data[3]));
            }else{
        // Return Not success
```

```
                }
            }
            return 0;
        }
};
```

Error/Return Code

```
0       // Success
-4      //User cancel (PINPAD)
-5      //Time out (PINPAD)
-10     //No PINPAD or PINPAD not available
-11     //PINPAD Bypass
1       //Parameter Error
2       //Object is Not Exist
3       //Object already exist
4       //Unknown Error
5       //System Error
6       //Function can't be executed
7       //Lack of memory
16      //Credential not exist
17      //No more parameter items exist
18      //Data Error
19      //The amount of data exceeds the agreed range.
20      //User Cancelled
21      //Unsupported Function
22      //Unsupported APP
23      //The current interface does not support
24      //There is no matching APP
27      //Duplicate TLV
29      //Card Data Length Error
30      //Script Syntax Error
31      //Script length exceeds 128 Bytes
32      //The card data element is missing (Mandatory item is missing)
33      //Card data is incorrect, existing but incorrect.
34      //Terminal data element missing (Mandatory item is missing)
35      //Terminal data is incorrect (existing but incorrect)
36      //Certificate error
37      //The CID in the card Response Data is inconsistent
38      //The card Response Data is not correct. (Format error, Data missing, Length error,etc.)
39      //Card locked
40      //Application in use condition is not satisfied
41      //CA public key not exist
42      //The certificate has been revoked
```

43 //No AID parameters downloaded
44 //The device has not been properly initialized

Return code for transaction process control:

50 //Ask for online PIN
51 //Go to transaction finish (not suspend the transaction)
52 //Suspend the transaction
53 //Go to online process (return in GAC)
56 //Transaction approved
57 //Transaction declined
58 //Execution fail
59 //Service is not allowed
60 //Ask for notification, but cannot go online
64 //Need to display the previous transaction record
65 //Need to display the next transaction record
68 //GPO return 0x6985, shall re-select AID
72 //Offline PIN Checking Error(can't get random number, the response for APDU is not 0x9000)
74 //Screen rotation event

Hardware Error:

80 //Can't detect Card in slot
81 //Card illegal
82 //Can't Detect Card
83 //PINPAD handling Error
//84 //Screen Operation Error
85 //IC Slot Operation Error
86 //PICC Slot Operation Error
87 //Write FLASH Error
88 //Get Date error
89 //Operation Timeout
90 //Hardware error
91 // Detect multiple PICC cards
92 //PIN Bypass
93 //PIN length Error
94 //PIN Checking—System Busy
95 //Authentication Not success
96 //Firmware update--- CRC ERROR
97 //Firmware update--- Length ERROR
//98 //Firmware update ---Not initialized
99 //Firmware update-- Padding error
//100 //Firmware update--- CRC ERROR
101 //System Busy
102 //Dual-Control Key Error

104 //TMK not exist
105 //Working key not exist
106 //KCV not correct
107 //Parameter Error
108 //Variable data field length error
109 //Frame format error
110 //Execute exception
111 //Database Error
112 //PIN incorrect
113 //Communication error between application chip and security firmware
116 //No Printer
117 //Unknown command
118 //LRC not success
119 //Transaction Timeout
120 //Other Error
121 //Terminal locked
122 //This parameter is not supported
123 //Command Cancelled
124 //Memory full
125 //Function is not executable
130 //Untreated
131 //Card is expired.
138 //Lack of Paper
139 //Printer Overheat

//Security firmware error:
160 //Protect-Key Error
161 //Key Type error
162 //Key is not exist
163 //Key memory overflow
164 //Key Index not correct
165 //Key Value illegal
166 //Key Length error
167 //KVC is incorrect