

POIGeneralAPI

```
POIGeneralAPI poiGeneralAPI = POIGeneralAPI.getDefault();
```

setDateTime

Set SP time

```
void setDateTime(byte[] time){...}  
  
poiGeneralAPI.setDateTime(HexUtil.parseHex("220616102300")); // 2022.06.16  
10:23:00
```

Parameter	Type	Description
time	byte[]	The date and time value, for a total of 6 bytes of BCD code. For example : October 1,2013,13:05'10, it is expressed as: 131001130510

getDateTime

Get SP time

```
byte[] getDateTime(){...}  
  
String time = HexUtil.toHexString(poiGeneralAPI.getDateTime());  
Log.d(TAG, "Time: " + time); // DateTime: 220616171714
```

Return	Type	Description
time	byte[]	The date and time value, for a total of 6 bytes of BCD code. For example : October 1,2013,13:05'10, it is expressed as: 131001130510

getSpVersion

Get SP version number

```
String getSpVersion(){...}  
  
String spVer = poiGeneralAPI.getSpVersion();  
Log.d(TAG, "SpVer: " + spVer); // SpVer: SP_V1.01.014 K002F600 220521 PEDSTA=4
```

Return	Type	Description
ver	String	SP version number

getSdkVersion

Get SDK version number

```
String getSdkVersion(){...}  
  
String sdkVer = poiGeneralAPI.getSdkVersion();  
Log.d(TAG, "SdkVer: " + sdkVer); // SdkVer: 2.1.17.20220523
```

Return	Type	Description
ver	String	SDK version number

getApVersion

Get the Android system version

没有得到结果

```
String getApVersion(){...}  
  
String apVer = poiGeneralAPI.getApVersion();
```

Return	Type	Description
apVer	String	

getVersion

Get version by type

```
String getVersion(int verType){...}  
  
public static final int VERSION_TYPE_SP           = 0x01;  
public static final int VERSION_TYPE_MP           = 0x02;  
public static final int VERSION_TYPE_AP           = 0x03;  
public static final int VERSION_TYPE_HW           = 0x04;  
public static final int VERSION_TYPE_SDK           = 0x05;  
public static final int VERSION_TYPE_PSN          = 0x06;  
public static final int VERSION_TYPE_DSN          = 0x07;  
public static final int VERSION_TYPE_DSN2         = 0x08;  
public static final int VERSION_TYPE_KERNEL       = 0x09;  
public static final int VERSION_TYPE_FLASHID      = 0x0a;  
public static final int VERSION_TYPE_CUSTOMER_NAME = 0x0b;  
public static final int VERSION_TYPE_CUSTOMER_SUBNAME = 0x0c;  
public static final int VERSION_TYPE_EXTERNAL_BASE = 0xff00;  
  
String ver = poiGeneralAPI.getVersion(POIGeneralAPI.VERSION_TYPE_KERNEL);
```

Parameter	Type	Description
verType	int	Version type

Return	Type	Description
ver	String	Corresponding version number

setLed

Turn On/Off LED indicator

```
void setLed(int color, boolean onOff){...}

public static final int LED_BLUE    = 0x10;
public static final int LED_GREEN   = 0x20;
public static final int LED_YELLOW  = 0x40;
public static final int LED_RED     = 0x80;

poiGeneralAPI.setLed(POIGeneralAPI.LED_BLUE, true);
```

Parameter	Type	Value	Constant	Description
color	int	0x80	LED_RED	Led red
		0x40	LED_YELLOW	Led yellow
		0x20	LED_GREEN	Led green
		0x10	LED_BLUE	Led blue
onOff	boolean	true false	-/-	On Off

setLedFlash

Set LED blink

```
void setLedFlash(int color, int onMs, int offMs){...}

poiGeneralAPI.setLedFlash(POIGeneralAPI.LED_BLUE, 1000, 1000);
```

Parameter	Type	Description
color	int	Same as color of setLed
onMs	int	The period of LED on, in milliseconds
offMs	int	The period of LED off, in milliseconds

setBeep

Turn On/Off beeper

```
void setBeep(boolean onOff, int freq, int timeMs){...}

poiGeneralAPI.setBeep(true, 1000, 500);
```

Parameter	type	Description
-----------	------	-------------

Parameter	type	Description
onOff	boolean	true ---- On false ---- Off
frep	int	Frequency in HZ
timeMs	int	Keep time in millisecond

setRFRegister

```
void setRFRegister(int cardType, int regAddr, int val){...}
```

Parameter	Type	Value	Conatant	Description
catdType	int			
regAddr	int			
val	int			

getRFRegister

```
int getRFRegister(byte[] data, PosByteArray rspBuf){...}
```

Parameter	Type	Description
data	byte[]	
rspBuf	PosByteArray	

Return	Type	Description
result	int	0 means success , others are failures

getAttackedLogs

```
int getAttackedLogs(byte[] data, PosByteArray rspBuf){...}
```

Parameter	Type	Description
-----------	------	-------------

Parameter	Type	Description
data	byte[]	
rspBuf	PosByteArray	

Return	Type	Description
result	int	0 means success , others are failures

transmitRawCmd

```
int transmitRawCmd(boolean sync, byte[] cmdBytes, PosByteArray rspBuf){...}
```

Parameter	Type	Description
sync	boolean	
cmdBytes	byte[]	
rspBuf	PosByteArray	

Return	Type	Description
result	int	0 means success , others are failures

getRFRegisterValue

```
int getRFRegisterValue(int cardType, int address){...}
```

Parameter	Type	Description
cardType	int	
address	int	

Return	Type	Description
val	int	

setRFRegisterRaw

```
int setRFRegisterRaw(byte[] tlvBytes){...}
```

Parameter	Type	Description
tlvBytes	byte[]	

Return	Type	Description
result	int	0 means success , others are failures

setRFTempRegisterRaw

```
int setRFTempRegisterRaw(byte[] tlvBytes){...}
```

Parameter	Type	Description
tlvBytes	byte[]	

Return	Type	Description
result	int	0 means success , others are failures

getRFTempRegisterValueRaw

```
int getRFTempRegisterValueRaw(byte[] inBytes, PosByteArray rspBuf){...}
```

Parameter	Type	Description
inBytes	byte[]	
rspBuf	PosByteArray	

Return	Type	Description
result	int	0 means success , others are failures

setParameters

```
int setParameters(byte[] tlvBytes, PosByteArray rspBuf){...}
```

Parameter	Type	Description
tlvBytes	byte[]	
rspBuf	PosByteArray	

Return	Type	Description
result	int	0 means success , others are failures

setProperty

Set system prop, only specific props are supported

```
int setProperty(String prop, String val){...}
```

Parameter	Type	Description
prop	String	Support : sys.pos.sub_sdk_version sys.pos.main_sdk_version
val	String	

Return	Type	Description
result	int	0 means success , others are failures

getSystemInfo

```
int getSystemInfo(byte[] tlvBytes, PosByteArray rspBuf){...}  
int getSystemInfo(int tag, PosByteArray rspBuf){...}
```

Parameter	Type	Description
tlvBytes	byte[]	
tag	int	

Parameter	Type	Description
rspBuf	PosByteArray	

Return	Type	Description
result	int	0 means success , others are failures

setScannerLed

```
int setScannerLed(boolean onOff){...}

poiGeneralAPI.setScannerLed(false);
```

Parameter	Type	Description
onOff	boolean	

Return	Type	Description
result	int	0 means success , others are failures

getSpVersionEx

```
String getSpVersionEx(int spId){...}
```

Parameter	Type	Description
spId	int	

Return	Type	Description
ver	String	

setParametersEx

```
int setParametersEx(int spId, byte[] tlvBytes, PosByteArray rspBuf){...}
```

Parameter	Type	Description
-----------	------	-------------

Parameter	Type	Description
spld	int	
tlvBytes	byte[]	
rspBuf	PosByteArray	

Return	Type	Description
result	int	0 means success , others are failures

setRFTempRegister

```
int setRFTempRegister(int cardType, int address, int value){...}
```

Parameter	Type	Description
cardType	int	
address	int	
value	int	

Return	Type	Description
result	int	0 means success , others are failures

getRFTempRegisterValue

```
int getRFTempRegisterValue(int cardType, int address){...}
```

Parameter	Type	Description
cardType	int	
address	int	

Parameter	Type	Description
value	int	

getAntennaPerformance

Get NFC Antenna Performance

```
int getAntennaPerformance(PosByteArray rspBuf){...}  
  
PosByteArray a2 = new PosByteArray();  
int ret = poiGeneralAPI.getAntennaPerformance(a2);  
Log.d(TAG, "Antenna performance: result= " + ret + " " + a2.toString());  
// Antenna performance: result= 0 PosByteArray:len= 12, buffer=  
01040000009d0204000000ff
```

Parameter	Type	Description
rspBuf	PosByteArray	Antenna Performance result Tlv Format : 01040000009d 0204000000ff 01 : amplitude 02 : phase

Return	Type	Description
result	int	0 means success , others are failures

setApplicationScene

Set the SDK scene. This flag can be update from lower level to higher level in only single direction and can not switch back from higher level to lower level.

```
int setApplicationScene(int appScene){...}  
  
public static final int PAYMENT_APPLICATION_SCENE_UNENCRYPTED    = 0x00;  
public static final int PAYMENT_APPLICATION_SCENE_MAG_ENCRYPTED = 0x01;  
public static final int PAYMENT_APPLICATION_SCENE_ENCRYPTED     = 0x02;  
  
int ret =  
poiGeneralAPI.setApplicationScene(POIGeneralAPI.PAYMENT_APPLICATION_SCENE_ENCRYP  
TED);
```

Parameter	Type	Value	Constant
appScene	int	0	PAYMENT_APPLICATION_SCENE_UNENCRYPTED
		1	PAYMENT_APPLICATION_SCENE_MAG_ENCRYPTED
		2	PAYMENT_APPLICATION_SCENE_ENCRYPTED

Return	Type	Description
result	int	0 means success , others are failures

getCurrentApplicationScene

Get the SDK scene

```
int getCurrentApplicationScene(){...}

int appScene = poiGeneralAPI.getCurrentApplicationScene();
Log.d(TAG, "app scene= " + appScene); // app scene= 0
```

Return	Type	Description
appScene	int	Same as appScene of setApplicationScene

interface POIGeneralAPI.EventListener

```
public interface EventListener {
    public void onInfo(POIGeneralAPI accessoryMgr, int what, int extra);

    public void onError(POIGeneralAPI accessoryMgr, int what, int extra);

    public void onTransmitRawCmdRet(POIGeneralAPI accessoryMgr, byte[]
rspBytes);
}

POIGeneralAPI.EventListener listener = new POIGeneralAPI.EventListener() {
    @Override
    public void onInfo(POIGeneralAPI poiGeneralAPI, int i, int i1) {...}

    @Override
    public void onError(POIGeneralAPI poiGeneralAPI, int i, int i1) {...}

    @Override
    public void onTransmitRawCmdRet(POIGeneralAPI poiGeneralAPI, byte[] bytes)
{...}
};
```

Interface	Description
onInfo	Called to indicate an info or a warning
onError	Called to indicate an error
onTransmitRawCmdRet	Called when the accessory module response

Parameter	Type	Description
accessoryMgr	POIGeneralAPI	Instance of POIGeneralAPI
what	int	the type of info or warning

Parameter	Type	Description
extra	int	an extra code, specific to the info. Typically implementation dependent.
rspBytes	byte[]	

registerListener

Register listener

```
void registerListener(EventListener listener){...}

poiGeneralAPI.registerListener(listener);
```

Parameter	Type
listener	POIGeneralAPI.EventListener

unregisterListener

Unregister listener

```
void unregisterListener(EventListener listener){...}

poiGeneralAPI.unregisterListener(listener);
```

Parameter	Type
listener	POIGeneralAPI.EventListener

class PosByteArray

```
public class PosByteArray implements Parcelable {
    public int len;
    public byte[] buffer;

    public PosByteArray() {
        len = 0;
        buffer = null;
    }
    public PosByteArray(int length) {
        len = length;
        buffer = new byte[length];
    }
    public PosByteArray(byte[] buf, int length) {
        len = length;
        buffer = buf;
    }

    public String toString() {...}
}
```

Attribute	Type	Description
len	int	response data length
buffer	byte[]	response data

POIPrinterManager

```

public class Printer {
    private static final String TAG = "Printer";

    public static void printerTest(Context context) {
        final POIPrinterManager printerManager = new POIPrinterManager(context);
        printerManager.open();
        int state = printerManager.getPrinterState();
        Log.d(TAG, "printer state = " + state);
        //printerManager.setPrintFont("/system/fonts/NotoSansTamil-Bold.otf");
        printerManager.setPrintGray(2000);
        printerManager.setLineSpace(5);
        //printerManager.cleanCache();
        String str1 = "This is an example of a receipt";
        PrintLine p1 = new TextPrintLine(str1, PrintLine.CENTER);
        printerManager.addPrintLine(p1);
        Bitmap bitmap = BitmapFactory.decodeResource(context.getResources(),
R.drawable.shopping_mall);
        printerManager.addPrintLine(new BitmapPrintLine(bitmap,
PrintLine.CENTER));
        printerManager.setPrintFont("/system/fonts/ComingSoon.ttf");
        String str2 = "Floor ** , Building **, No.*** LONG DONG Avenue, Pudong
New District, Shanghai, China";
        PrintLine p2 = new TextPrintLine(str2, PrintLine.LEFT, 20);
        printerManager.addPrintLine(p2);
        printerManager.setPrintFont("/system/fonts/DroidSansMono.ttf");
        List<TextPrintLine> list1 = printList("24 June 2025", "    Assistant
6", "815002", 18, false);
        printerManager.addPrintLine(list1);
        List<TextPrintLine> list2 = printList("Item", "Quantity", "Price", 24,
true);
        printerManager.addPrintLine(list2);
        List<TextPrintLine> list3 = printList("Tomato", "1", "$2.08", 24,
false);
        printerManager.addPrintLine(list3);
        List<TextPrintLine> list4 = printList("Orange", "1", "$1.06", 24,
false);
        printerManager.addPrintLine(list4);
        PrintLine p3 = new TextPrintLine("Total $3.14", PrintLine.RIGHT);
        printerManager.addPrintLine(p3);
        printerManager.addPrintLine(new TextPrintLine(""));
        // drawable/barcode.jpg 100*100
        bitmap = BitmapFactory.decodeResource(context.getResources(),
R.drawable.barcode);
        printerManager.addPrintLine(new BitmapPrintLine(bitmap,
PrintLine.CENTER));
        printerManager.addPrintLine(new TextPrintLine(""));
    }
}

```

```

        String str3 = "Did you know you could have earned Rewards points on this
purchase?";
        PrintLine p4 = new TextPrintLine(str3, PrintLine.CENTER);
        printerManager.addPrintLine(p4);
        PrintLine p5 = new TextPrintLine("Simply sign up today for a Membership
Card!", PrintLine.CENTER);
        printerManager.addPrintLine(p5);
        bitmap = BitmapFactory.decodeResource(context.getResources(),
R.drawable.barcode_12345);
        printerManager.addPrintLine(new BitmapPrintLine(bitmap,
PrintLine.CENTER));
        printerManager.addPrintLine(new TextPrintLine(" ", 0, 100));
        POIPrinterManager.IPrinterListener listener = new
POIPrinterManager.IPrinterListener() {
            @Override
            public void onStart() {}

            @Override
            public void onFinish() {
                printerManager.close();
            }

            @Override
            public void onError(int code, String msg) {
                Log.e("POIPrinterManager", "code: " + code + "msg: " + msg);
                printerManager.close();
            }
        };
        if(state == 4){
            printerManager.close();
            return;
        }
        printerManager.beginPrint(listener);
    }

    private static List<TextPrintLine> printList(String leftStr, String
centerStr, String rightStr, int size, boolean bold){
        TextPrintLine textPrintLine1 = new TextPrintLine(leftStr,
PrintLine.LEFT, size, bold);
        TextPrintLine textPrintLine2 = new
TextPrintLine(centerStr,PrintLine.CENTER, size, bold);
        TextPrintLine textPrintLine3 = new TextPrintLine(rightStr,
PrintLine.RIGHT, size, bold);
        List<TextPrintLine> list = new ArrayList<>();
        list.add(textPrintLine1);
        list.add(textPrintLine2);
        list.add(textPrintLine3);
        return list;
    }
}

```

open

Open printer

```
void open(){...}

printerManager.open();
```

close

Close printer

```
void close(){...}

printerManager.close();
```

getPrinterState

Get printer state

```
int getPrinterState(){...}

public static final int STATUS_IDLE      = 0;
public static final int STATUS_PRINTING  = 1;
public static final int STATUS_OVERHEAT  = 2;
public static final int STATUS_NO_PAPER  = 3;
public static final int STATUS_NO_PRINTER = 4;

int state = printerManager.getPrinterState();
Log.d(TAG, "printer state = " + state); // printer state = 4
```

Return	Type	Value	Constant	Description
state	int	0	STATUS_IDLE	idle
		1	STATUS_PRINTING	printing
		2	STATUS_OVERHEAT	over heat
		3	STATUS_NO_PAPER	no paper
		4	STATUS_NO_PRINTER	no printer

setPrintGray

Set the print grayscale

```
void setPrintGray(int gray){...}

printerManager.setPrintGray(2000);
```

Parameter	Type	Default	Description
-----------	------	---------	-------------

Parameter	Type	Default	Description
gray	int	1200	The value smaller the lighter, the larger the thicker gray <code><= 0</code> , not effective get value: <code>gray & 0xffff</code> It is recommended to adjust in units of 200

setPrintFont

Set the print font

```
void setPrintFont(String path){...}

printerManager.setPrintFont("/system/fonts/DroidSansMono.ttf");
```

Parameter	Type	Description
path	String	System font path <code>"/system/fonts/....."</code>

setLineSpace

Set line space

```
void setLineSpace(int line){...}

printerManager.setLineSpace(5);
```

Parameter	Type	Description
line	int	line space Reference <code>textView.setLineSpacing(line, 1.0f);</code>

cleanCache

clean cache. It is used for multiple [beginPrint](#) between [open](#) and [close](#)

```
void cleanCache(){...}

printerManager.cleanCache();
```

addPrintLine

add print content

```
void addPrintLine(PrintLine line){...}
void addPrintLine(List<TextPrintLine> line){...}

printerManager.addPrintLine(new TextPrintLine("123", PrintLine.CENTER, 50));
```


Parameter	Type	Description
line	PrintLine <code>List<TextPrintLine></code>	TextPrintLine BitmapPrintLine <code>List<TextPrintLine></code>

class PrintLine

```
public class PrintLine {
    public static final int TEXT    = 0;
    public static final int BITMAP  = 1;

    public static final int LEFT    = 0;
    public static final int CENTER  = 1;
    public static final int RIGHT   = 2;

    protected int type;
    protected int position;
}
```

class TextPrintLine

```
public class TextPrintLine extends PrintLine {
    public static final int FONT_SMALL  = 16;
    public static final int FONT_NORMAL = 24;
    public static final int FONT_LARGE  = 36;

    public TextPrintLine() {...}
    public TextPrintLine(String content) {...}
    public TextPrintLine(String content, int position) {...}
    public TextPrintLine(String content, int position, int size) {...}
    public TextPrintLine(String content, int position, int size, boolean bold)
    {...}
    public TextPrintLine(String content, int position, int size, boolean bold,
    boolean italic) {...}
    public TextPrintLine(String content, int position, int size, boolean bold,
    boolean italic, boolean invert) {...}
}

new TextPrintLine("1234567890", PrintLine.CENTER, 24, false, false, true);
```

Attribute	Type	Value	Default	Description
type	int	PrintLine.TEXT	-/-	-/-
content	String	-/-	-/-	print content
position	int	PrintLine.LEFT PrintLine.RIGHT PrintLine.CENTER	PrintLine.LEFT	Align left Align right Align center
size	int	-/-	TextPrintLine.FONT_NORMAL	font size

Attribute	Type	Value	Default	Description
bold	boolean	true false	false	Whether the printed text is bold
italic	boolean	true false	false	Whether the printed text is italicized
invert	boolean	true false	false	Black characters on white background White characters on black background

class BitmapPrintLine

```
public class BitmapPrintLine extends PrintLine {
    private Bitmap bitmap;

    public BitmapPrintLine() {...}
    public BitmapPrintLine(Bitmap bitmap) {...}
    public BitmapPrintLine(Bitmap bitmap, int position) {...}

    public Bitmap getBitmap() {...}
    public void setBitmap(Bitmap bitmap) {...}
}

// drawable/test.jpg
Bitmap bitmap = BitmapFactory.decodeResource(context.getResources(),
R.drawable.test);
printerManager.addPrintLine(new BitmapPrintLine(bitmap, PrintLine.CENTER));
```

Attribute	Type	Value	Default	Description
type	int	PrintLine.BITMAP	-/-	-/-
bitmap	Bitmap	-/-	null	-/-
position	int	PrintLine.LEFT PrintLine.RIGHT PrintLine.CENTER	PrintLine.LEFT	Align left Align right Align center

beginPrint

start to print

```
void beginPrint(IPrinterListener printerListener){...}

printerManager.beginPrint(listener);
```

Parameter	Type	Description
-----------	------	-------------

Parameter	Type	Description
printerListener	IPrinterListener	print listener

interface IPrinterListener

```
public interface IPrinterListener {
    void onStart();

    void onFinish();

    void onError(int code, String msg);
}

public static final int ERROR_INIT      = -1;
public static final int ERROR_PRINT     = -2;
public static final int ERROR_OVERHEAT = -3;
public static final int ERROR_NO_PAPER = -4;
public static final int ERROR_OTHER    = -999;

POIPrinterManager.IPrinterListener listener = new
POIPrinterManager.IPrinterListener() {
    @Override
    public void onStart() {}

    @Override
    public void onFinish() {
        printerManager.close();
    }

    @Override
    public void onError(int code, String msg) {
        Log.e("POIPrinterManager", "code: " + code + "msg: " + msg);
        printerManager.close();
    }
};
```

Interface	Description
onStart	Called when printing starts
onFinish	Called at the end of printing
onError	Called to indicate an error

Parameter	Type	Value	Decription
code	int	POIPrinterManager.ERROR_INIT = -1 POIPrinterManager.ERROR_PRINT = -2 POIPrinterManager.ERROR_OVERHEAT = -3 POIPrinterManager.ERROR_NO_PAPER = -4 POIPrinterManager.ERROR_OTHER = -999	error code

Parameter	Type	Value	Description
msg	String	Printer init failed. In the case of a lock screen, printing is not support. Printer status failed. Printer bitmap failed. Printer overheat. Printer no paper. Printer other error.	ERROR_INIT ERROR_PRINT ERROR_PRINT ERROR_PRINT ERROR_OVERHEAT ERROR_NO_PAPER ERROR_OTHER

lineWrap

Print n blank lines at the beginning

```
void lineWrap(int value){...}

printerManager.lineWrap(3);
```

Parameter	Type	Description
value	int	n lines, fixed length

getPrinterLength

Total length of printed data

```
double getPrinterLength(){...}

double len = printerManager.getPrinterLength();
```

Return	Type	Description
len	double	Total length printed between open and close

getBeforePrinterLength

Total length of data printed before this print

```
double getBeforePrinterLength(){...}

double lenb = printerManager.getBeforePrinterLength();
```

Return	Type	Description
len	double	Current print length: <code>len - lenb</code> Except for the first print First print: <code>len == lenb</code>

POIHsmManage

```
POIHsmManage hsmManage = POIHsmManage.getDefault();
```

PedCalDes

Data encryption and decryption

```
int PedCalDes(int tdkIdx, int mode, byte[] inBuf, byte[] ivBytes, PosByteArray
rspBuf){...}
int PedCalDes(int tdkIdx, int mode, byte[] inBuf, PosByteArray rspBuf){...}

public static final int PED_CALC_DES_MODE_ECB_DEC      = 0x00;
public static final int PED_CALC_DES_MODE_ECB_ENC      = 0x01;
public static final int PED_CALC_DES_MODE_CBC_DEC      = 0x02;
public static final int PED_CALC_DES_MODE_CBC_ENC      = 0x03;
public static final int PED_CALC_DES_MODE_AES_ECB_DEC  = 0x10;
public static final int PED_CALC_DES_MODE_AES_ECB_ENC  = 0x11;
public static final int PED_CALC_DES_MODE_AES_CBC_DEC  = 0x12;
public static final int PED_CALC_DES_MODE_AES_CBC_ENC  = 0x13;
public static final int PED_CALC_DES_MODE_SM4_ECB_DEC  = 0x20;
public static final int PED_CALC_DES_MODE_SM4_ECB_ENC  = 0x21;
public static final int PED_CALC_DES_MODE_SM4_CBC_DEC  = 0x22;
public static final int PED_CALC_DES_MODE_SM4_CBC_ENC  = 0x23;

String data = "95623658568800013865013D00001209";
byte[] dataIn = HexUtil.parseHex(data); // data.length % 8 == 0
byte[] dataOut = new byte[128];
PosByteArray rspBuf = new PosByteArray(dataOut, dataOut.length);
int ret = hsmManage.PedCalDes(1, 3, dataIn, rspBuf);
```

Parameter	Type	Value	Description
tdkIdx	int	1~64	index of TDK , need write TDK
mode	int	0x00、0x01、0x02、 0x03 0x10、0x11、0x12、 0x13 0x20、0x21、0x22、 0x23	DEC ---- decryption ENC ---- encryption
inBuf	byte[]	-/-	Data to be encrypted/decryption
ivBytes	byte[]		
rspBuf	PosByteArray	-/-	Processed data

Return	Type	Description
result	int	0 means success , others are failures

PedWriteKey

Write key into POS

```
int PedWriteKey(PedKeyInfo keyInfo, PedKcvInfo kcvInfo){...}

int WriteKey(int srcKeyType, int srcKeyIndex, int dstKeyType, int dstKeyIndex,
int alghFlag, int dstKeyLen, byte[] bufIn, int mode, byte[] aucCheckBufIn) {
    byte[] kcv;
    if (mode == 0) {
        kcv = new byte[5];
    } else {
        kcv = new byte[aucCheckBufIn.length + 1];
        kcv[0] = (byte) aucCheckBufIn.length;
        System.arraycopy(aucCheckBufIn, 0, kcv, 1, aucCheckBufIn.length);
    }
    PedKeyInfo pedKeyInfo = new PedKeyInfo(srcKeyType, srcKeyIndex, dstKeyType,
dstKeyIndex, alghFlag, dstKeyLen, bufIn);
    PedKcvInfo pedKcvInfo = new PedKcvInfo(mode, kcv);
    return hsmManage.PedWriteKey(pedKeyInfo, pedKcvInfo);
}

// 11111111111111111111111111111111 TMK
byte[] key = HexUtil.parseHex("F40379AB9E0EC533F40379AB9E0EC533");
byte[] kcv = HexUtil.parseHex("82E13665");
int result = WriteKey(POIHsmManage.PED_TMK, 1, POIHsmManage.PED_TDK, 1, 0,
key.length, key, 1, kcv);
if (result == 0) {
    Log.e(TAG, "load TDK success!");
} else {
    Log.e(TAG, "error " + result);
}
```

Parameter	Type	Description
keyInfo	PedKeyInfo	Information about the key to write
kcvInfo	PedKcvInfo	Key check value

Return	Type	Description
result	int	0 means success , others are failures

class PedKeyInfo

Information about the key to write

```
public class PedKeyInfo implements Parcelable
{
    public int srcKeyType;
    public int srcKeyIdx;
    public int dstKeyType;
    public int dstKeyIdx;
    public int dstAlgorithm;
    public int dstKeyLen;
}
```

```

public byte[] dstKeyData;
public byte[] reserved;
public long writeKeyTimeMs;
public String callingPkgName;

public PedKeyInfo() {...}
public PedKeyInfo(int srcKeyType, int srcKeyIdx, int dstKeyType,
    int dstKeyIdx, int dstAlgorithm, int dstKeyLen, byte[] dstKeyData) {...}
public PedKeyInfo(int srcKeyType, int srcKeyIdx, int dstKeyType,
    int dstKeyIdx, int dstAlgorithm, int dstKeyLen, byte[] dstKeyData,
    long timeMs, String pkgName) {...}
public PedKeyInfo(PedKeyInfo o) {...}
}

public static final int PED_TLK = 0x01;
public static final int PED_TMK = 0x02;
public static final int PED_TPK = 0x03;
public static final int PED_TAK = 0x04;
public static final int PED_TDK = 0x05;
public static final int PED_TEK = 0x06;
public static final int PED_TTK = 0x09;
public static final int PED_TIK = 0x10;
public static final int PED_TRK = 0x11;

```

Attribute	Type	Value	Constant	Description
srcKeyType	int	0 1 2	-/- PED_TLK PED_TMK	Plaintext key TLK encryption TMK encryption
srcKeyIdx	int	0 1 1~64	-/-	Plaintext key TLK : Only 1 group is supported TMK : Support 64 groups
detKeyType	int	1 2 3 4 5 6 9 16 17	PED_TLK PED_TMK PED_TPK PED_TAK PED_TDK PED_TEK PED_TTK PED_TIK PED_TRK	Type of Target Key (Type of key to write)
detKeyIdx	int	1 1~64	-/-	Index of Target Key TLK only support 1 group Other key support 64 groups
dstAlgorithm	int	0 1	-/-	Encryption algorithm of Target Key 0 ---- TDES 1 ---- AES
dstKeyLen	int	8/16/24/32	-/-	Target Key length

Attribute	Type	Value	Constant	Description
dstKeyData	byte[]	-/-	-/-	Target key data Plaintext or ciphertext
reserved	byte[]	byte[10]	-/-	Reserved space
writeKeyTimeMs	long	-/-	-/-	
callingPkgName	String	-/-	-/-	

class PedKcvInfo

Key check value

```
public class PedKcvInfo implements Parcelable {
    public int checkMode;
    public byte[] checkBuf;

    public PedKcvInfo(){...}
    public PedKcvInfo(int mode, byte[] buffer){...}
    public PedKcvInfo(PedKcvInfo o){...}
}
```

Attribute	Type	Description
checkMode	int	Mode 0 : the value of checkBuf is invalid, KCV is not need to verify, checkBuf can be invalid data Mode 1 : the checkBuf[0]=4, and one block of 0x00 encrypted by destination key, the first 4 bytes of result are KCV Mode 2 : Performs odd parity on the destination key and then checks by Mode 1 Mode 3 : Performs parity check on the destination key, and then checks by Mode 1
checkBuf	byte[]	KCV valid checkBuf[0] is length, the following is the data

PedWriteTIK

Write DUKPT Initial Key

```
int PedwriteTIK(int grpIdx, int t1Idx, int tikLen, byte[] tikData, byte[] ksn,
PedKcvInfo kcvInfo){...}

PedKcvInfo kcvInfo = new PedKcvInfo(0, new byte[5]);
String keyData = "0123456789ABCDEF0123456789ABCDEF";
byte[] key = HexUtil.parseHex(keyData);
String ksnData = "FFFF9080102495000001";
hsmManage.PedWriteTIK(1, 0, key.length, key, HexUtil.parseHex(ksnData),
kcvInfo);
```


Parameter	Type	Description
grpIdx	int	DUKPT key group index number Supports 10 groups, the index range is [1, 10]
tlkIdx	int	TLK index number
tikLen	int	TIK length n now DUKPT algorithm supports 8/16 bytes length key.
tikData	byte[]	Initial encryption key data. If TIKIdx is 0, this is clear-text. If TIKIdx is not 0, this is cipher-text encrypted by TLK.
ksn	byte[]	Initialize KSN
kcvInfo	PedKcvInfo	Key check value

Return	Type	Description
result	int	0 means success , others are failures

PedGetPinBlock

Get PIN-BLOCK cipher-text.

```
int PedGetPinBlock(int fetchMode, int idxOrGrpId, int fmt, int timeoutMs, byte[]
data, String expPinLenInd){...}

public static final int PED_PINBLOCK_FETCH_MODE_TPK    = 1;
public static final int PED_PINBLOCK_FETCH_MODE_DUKPT = 2;

public static final int PED_PINBLOCK_TPK_FMT_ISO9564_0 = 0x00;
public static final int PED_PINBLOCK_TPK_FMT_ISO9564_1 = 0x01;
public static final int PED_PINBLOCK_TPK_FMT_ISO9564_2 = 0x02;
public static final int PED_PINBLOCK_TPK_FMT_HKEPS     = 0x03;

public static final int PED_PINBLOCK_DUKPT_FMT_ISO9564_0_KSN_INC = 0x00;
public static final int PED_PINBLOCK_DUKPT_FMT_ISO9564_1_KSN_INC = 0x01;
public static final int PED_PINBLOCK_DUKPT_FMT_ISO9564_2_KSN_INC = 0x02;
public static final int PED_PINBLOCK_DUKPT_FMT_HKEPS_KSN_INC     = 0x03;
public static final int PED_PINBLOCK_DUKPT_FMT_ISO9564_0         = 0x20;
public static final int PED_PINBLOCK_DUKPT_FMT_ISO9564_1         = 0x21;
public static final int PED_PINBLOCK_DUKPT_FMT_ISO9564_2         = 0x22;
public static final int PED_PINBLOCK_DUKPT_FMT_HKEPS             = 0x23;

int keyMode = POIHsmManage.PED_PINBLOCK_FETCH_MODE_TPK;
int fmt = POIHsmManage.PED_PINBLOCK_TPK_FMT_ISO9564_0;
byte[] data = new byte[24];
String pinCard = "9704079989561273";
String pinCardEnc = "0000407998956127";
byte[] temp = pinCardEnc.getBytes();
System.arraycopy(temp, 0, data, 0, 16);
byte[] formatData = {0, 0, 0, 0, 0, 0, 0, 0, 0};
System.arraycopy(formatData, 0, data, 16, 8);
```

```
hsmManage.PedGetPinBlock(keyMode, 1, fmt, 30000, data,
"0,4,5,6,7,8,9,10,11,12");
```

Paremeter	Type	Description
fetchMode	int	Key management type: 0x01: Get Pin Block using MK pin key. 0x02: Get Pin Block using DUKPT pin key.
idxOrGrpId	int	detKeyIdx of PedKeyInfo or grpIdx of PedWriteTIK
fmt	int	Format of PIN BLOCK. If MK: 0x00 - ISO9564 format 0 0x01 - ISO9564 format 1 0x02 - ISO9564 format 3 0x04 - ISO9564 format 4 If DUKPT: 0x00 - ISO9564 format 0 KSN auto-increment. 0x01 - ISO9564 format 1 KSN auto-increment. 0x02 - ISO9564 format 2 KSN auto-increment. 0x20 - ISO9564 format 0 KSN not auto-increment. 0x21 - ISO9564 format 1 KSN not auto-increment. 0x22 - ISO9564 format 2 KSN not auto-increment.
timeoutMs	int	Timeout (ms), 0: no timeout, PED do not do timeout control.
data	byte[]	Consisting of two parts. The first 16 bytes: For ISO9564 format 0/1/3: The card number without check digit, if the account number is less than 16 characters fill '0' at the left of card number. For ISO9564 format 4: The first byte (byte[0]) is the number of PAN digits. The PAN digits is filled into the bytes array start from byte[1] in BCD form, every byte contain 2 BCD digits. The last 8 bytes are: Participate in Pin Block formatted 8-byte data (according to ISO9564 specification, the data can be random number, transaction serial number or timestamp, etc., but the upper 4 bits and lower 4 bits of each byte must be 0xA~0xF between.
expPinLenInd	String	Expect PIN length list. For example, supports length of pins is 4/6/8 and support pressing the enter key without entering a password. This string should be "0, 4, 6, 8", The '0' means allowed press the enter key directly. PIN length is up to 12 digits.

Return	Type	Description
--------	------	-------------

Return	Type	Description
result	int	0 means success , others are failures

PedTestPinBlock

```
int PedTestPinBlock(){...}

hsmManage.PedTestPinBlock();
```

PedGetMac

Calculate MAC of data input (using MK MAC_KEY)

```
int PedGetMac(int takIdx, int operationMode, byte[] data, byte[] ivBytes,
PosByteArray rspBuf){...}
int PedGetMac(int takIdx, int operationMode, byte[] data, PosByteArray rspBuf)
{...}
```

Parameter	Type	Description
takIdx	int	Index of MAK key in MK (TAK) , detKeyIdx of PedKeyInfo
operationMode	int	Operation mode: 0x00: CBC-MAC Use MAC-Key to encrypt message in CBC mode, the last block of cipher-text is the MAC result. 0x01: XOR-ECB-MAC Do xor operation with each block of message, use MAC-Key encrypt the finally XOR result. 0x02: ANSI-X9.19 MAC 0x03: ANSI-X9.9 MAC
data	byte[]	Message input to calculate MAC
ivBytes	byte[]	
rspBuf	PosByteArray	MAC output

Return	Type	Description
result	int	0 means success , others are failures

PedGetMacDukpt

Calculate MAC of data input (using DUKPT MAC_KEY)

```
int PedGetMacDukpt(int grpIdx, int operationMode, byte[] data, byte[] ivBytes,
PosByteArray macBuf, PosByteArray ksnBuf){...}
int PedGetMacDukpt(int grpIdx, int operationMode, byte[] data, PosByteArray
macBuf, PosByteArray ksnBuf){...}
```

Parameter	Type	Description
grpIdx	int	DUKPT group id, grpIdx of PedWriteTIK
operationMode	int	MAC operation control code: This code is coded by two part: X + Y. E.g 21 X can be: 00 - Use "both ways variant", KSN auto-increment. 20 - Use "both ways variant", KSN not auto-increment. 40 - Use "Response variant", KSN not auto-increment. Note: the 20 and 40 is decimal number. Y can be: 0 - CBC-MAC 1 - XOR-ECB-MAC 2 - ANSI-X9.19 MAC
data	byte[]	Message input to calculate MAC.
ivBytes	byte[]	
macBuf	PosByteArray	MAC output
ksnBuf	PosByteArray	KSN output

Return	Type	Description
result	int	0 means success , others are failures

PedVerifyPlainPin

Offline PIN plain text verify

```
int PedVerifyPlainPin(int slot, int mode, int timeoutMs, String expPinLenInd)
{...}

hsmManage.PedVerifyPlainPin(0, 0, 30000, "0,4,5,6,7,8,9,10,11,12");
```

Parameter	Type	Description
slot	int	IC Card slot ID. Default 0
mode	int	Mode, fixed 0

Parameter	Type	Description
timeoutMs	int	Timeout in millisecond
expPinLenInd	String	Same as expPinLenInd of PedGetPinBlock

Return	Type	Description
result	int	0 means success , others are failures Offline Plain Text verification requires the user to enter the PIN, which must be executed asynchronously. All execution results are notified to the upper layer through the callback function: onPedVerifyPin , type 0x86

PedVerifyCipherPin

Offline Cipher Text PIN verify

```
int PedVerifyCipherPin(int slot, int mode, int timeoutMs, String expPinLenInd,
PedRsaPinKey rsaPinKey){...}

PedRsaPinKey rsaPinKey = new PedRsaPinKey(module, exponent, random);
hsmManage.PedVerifyCipherPin(0, 0, 30000, "0,4,5,6,7,8,9,10,11,12", rsaPinKey);
```

Parameter	Type	Description
slot	int	IC Card slot ID. Default 0
mode	int	Mode, fixed 0
timeoutMs	int	Timeout in millisecond
expPinLenInd	String	Same as expPinLenInd of PedGetPinBlock
rsaPinKey	PedRsaPinKey	-/-

Return	Type	Description
result	int	0 means success , others are failures Offline cipher text verification requires the user to enter the PIN, which must be executed asynchronously. All execution results are notified to the upper layer through the callback function: onPedVerifyPin , type 0x87

class PedRsaPinKey

```

public class PedRsaPinKey implements Parcelable {
    public byte[] modData;
    public byte[] expData;
    public byte[] iccRandomData;

    public PedRsaPinKey() {...}
    public PedRsaPinKey(byte[] modData, byte[] expData, byte[] iccRandomData)
    {...}
    public PedRsaPinKey(PedRsaPinKey o) {...}
}

```

Attribute	Type	Description
modData	byte[]	Public key modulus
expData	byte[]	Public key exponent
iccRandomData	byte[]	Random number from IC card

PedGetKcv

Get the KCV data of KEY

```

int PedGetKcv(int keyType, int keyIdx, PedKcvInfo kcvInfo, PosByteArray rspBuf)
{...}

```

Parameter	Type	Description
keyType	int	Key type
keyIdx	int	Key index
kcvInfo	PedKcvInfo	Key check value
rspBuf	PosByteArray	Kcv data output

Return	Type	Description
result	int	0 means success , others are failures

PedErase

Clear all MK/SK and DUKPT keys in POI device

```

int PedErase(){...}

hsmManage.PedErase();

```

Return	Type	Description
--------	------	-------------

Return	Type	Description
result	int	0 means success , others are failures

PedGetRsaKey

Generate RSA key

```
int PedGetRsaKey(int publicKeyIndex, int privateKeyIndex, int size){...}
```

Parameter	Type	Description
publicKeyIndex	int	Public key index
privateKeyIndex	int	Private key index
size	int	Key size

Return	Type	Description
result	int	0 means success , others are failures

PedWriteRsaKey

Write RSA key

```
int PedwriteRsaKey(int keyIndex, byte[] modulus, byte[] exponent){...}
```

Parameter	Type	Description
keyIndex	int	Key index
modulus	byte[]	Modulus
exponent	byte[]	Exponent

Return	Type	Description
result	int	0 means success , others are failures

PedReadRsaKey

Read RSA key

```
int PedReadRsaKey(int keyIndex, PosByteArray rspData, PosByteArray rspModulus, PosByteArray rspExponent){...}
```

Parameter	Type	Description
keyIndex	int	Key index
rspData	PosByteArray	RSA Info, Format : Byte1 Key Type : 0x00 : does not exist 0x01 : Public key 0x02 : Private key Byte2-3 RSA Key Number E.g: 0800 bit == 2048 bit == 256 byte
rspModulus	PosByteArray	Public Modulus
rspExponent	PosByteArray	Public Exponent

Return	Type	Description
result	int	0 means success , others are failures

PedCalcRsa

RSA encryption and decryption

```
int PedCalcRsa(int keyIndex, int mode, byte[] data, PosByteArray rspBuf){...}
```

Parameter	Type	Description
keyIndex	int	Key index
mode	int	Mode
data	byte[]	Input data
rspBuf	PosByteArray	Data output

Return	Type	Description
result	int	0 means success , others are failures

PedWriteProtectKey

```
int PedWriteProtectKey(int type, int keyType, int keyIndex, byte[] data, int writeType, int writeIndex){...}
```

Parameter	Type	Description
-----------	------	-------------

Parameter	Type	Description
type	int	
keyType	int	
keyIndex	int	
data	byte[]	
writeType	int	
writeIndex	int	

Return	Type	Description
result	int	0 means success , others are failures

PedDukptDes

Data encryption or decryption by DUKPT data-key

```
int PedDukptDes(int grpIdx, int keyType, byte[] initVector, int mode, byte[]
data, PosByteArray rspKsn, PosByteArray rspBuf){...}

public static final int PED_DUKPT_TYPE_MAC           = 0x00;
public static final int PED_DUKPT_TYPE_DATA_REQUEST = 0x01;
public static final int PED_DUKPT_TYPE_DATA_RESPONSE = 0x02;
public static final int PED_DUKPT_TYPE_PIN           = 0x03;

public static final int PED_CALC_DES_MODE_ECB_DEC    = 0x00;
public static final int PED_CALC_DES_MODE_ECB_ENC    = 0x01;
public static final int PED_CALC_DES_MODE_CBC_DEC    = 0x02;
public static final int PED_CALC_DES_MODE_CBC_ENC    = 0x03;

//0123456789ABCDEF0123456789ABCDEF    FFFF9080102495000001
String data = "95623658568800013865013D00001209";
byte[] dataIn = encryptDataPadding(HexUtil.parseHex(data));
//dataPadding(data).getBytes();
int ret;
PosByteArray rspBuf = new PosByteArray();
PosByteArray rspKsn = new PosByteArray();
ret = hsmManage.PedDukptDes(1, POIHsmManage.PED_DUKPT_TYPE_PIN, new byte[8],
POIHsmManage.PED_CALC_DES_MODE_ECB_ENC, dataIn, rspKsn, rspBuf);
Log.d(TAG, "ret = " + ret + "\nksn = " + rspKsn + "\ndata = " + rspBuf);
//ret = 0
//ksn = PosByteArray:len= 10, buffer= ffff9080102495000001
//data = PosByteArray:len= 16, buffer= 586bbb7ee4e6be7a3c636f117f1fd35a
```

Parameter	Type	Description
grpIdx	int	grpIdx of PedWriteTIK

Parameter	Type	Description
keyType	int	Key type: 0x00 - Use Request or both ways MAC key 0x01 - Use Request or both ways Data key 0x02 - Use DATA Response key 0x03 - Use PIN Encryption key
initVector	byte[]	8-byte initial vector, required for CBC encryption and decryption.
mode	int	Operation mode: 0x00 - ECB decryption 0x01 - ECB encryption 0x02 - CBC decryption 0x03 - CBC encryption
data	byte[]	Data input
rspKsn	PosByteArray	KSN output (10 bytes)
rspBuf	PosByteArray	Data output

Return	Type	Description
result	int	0 means success , others are failures

PedGetDukptKsn

Get the KSN of next DUKPT operation

```
int PedGetDukptKsn(int grpIdx, PosByteArray rspBuf){...}

PosByteArray rspKsn = new PosByteArray();
hsmManage.PedGetDukptKsn(1, rspKsn);
Log.d(TAG, "ksn = " + rspKsn);
//ksn = PosByteArray:len= 10, buffer= ffff9080102495000002
```

Parameter	Type	Description
grpIdx	int	grpIdx of PedWriteTIK
rspBuf	PosByteArray	KSN output (10 bytes)

Return	Type	Description
result	int	0 means success , others are failures

PedDukptIncreaseKsn

KSN+1, the DUKPT key for each KSN, can only be used up to 256 times.

When a single key is used 256 times, it will return error, user need to call this interface, to make KSN increase.

```
int PedDukptIncreaseKsn(int grpIdx){...}
```

```
hsmManage.PedDukptIncreaseKsn(1);
```

Parameter	Type	Description
grpIdx	int	grpIdx of PedWriteTIK

Return	Type	Description
result	int	0 means success , others are failures

PedGetRandom

Get random number compliance with NIST SP800-90A

```
int PedGetRandom(int randomLen, PosByteArray rspBuf){...}
```

```
PosByteArray rspBuf = new PosByteArray();
int ret = hsmManage.PedGetRandom(10, rspBuf);
Log.d(TAG, "ret = " + ret + " " + rspBuf);
// ret = 0 PosByteArray:len= 10, buffer= c53d067cb6b28b66bb21
```

Parameter	Type	Description
randomLen	int	Random number length
rspBuf	PosByteArray	Random number output

PedKeyManage

Password management

```
int PedKeyManage(int type){...}
```

```
public static final int PED_KEY_MANAGE_CMD_UNLOCK_TERMINAL_KEY      = 0x80;
public static final int PED_KEY_MANAGE_CMD_ADMIN_A_KEY              = 0x81;
public static final int PED_KEY_MANAGE_CMD_ADMIN_B_KEY              = 0x82;
public static final int PED_KEY_MANAGE_CMD_KEEP_STATE               = 0x83;
public static final int PED_KEY_MANAGE_CMD_UNLOCK_TERMINAL_KEY_NEW  = 0x84;
public static final int PED_KEY_MANAGE_CMD_ADMIN_A_KEY_NEW          = 0x85;
public static final int PED_KEY_MANAGE_CMD_ADMIN_B_KEY_NEW          = 0x86;
```

Parameter	Type	Description
type	int	Management type: 0x80 - Verify unlock password. 0x81 - Verify Admin-A password. 0x82 - Verify Admin-B password. 0x83 - Hold sensitive state. 0x84 - Set new unlock password. 0x85 - Set new Admin-A password. 0x86 - Set new Admin-B password.

Return	Type	Description
result	int	0 means success , others are failures

PedCancelPinBlock

Cancel PIN entering process

```
int PedCancelPinBlock(){...}

hsmManage.PedCancelPinBlock();
```

Return	Type	Description
result	int	0 means success , others are failures

SysHwSelfCheck

Hardware self-test

```
int SysHwSelfCheck(int hwDeviceMask){...}

public static final int HW_SELF_CHECK_MASK_MAG      = 0x01;
public static final int HW_SELF_CHECK_MASK_ICC      = 0x02;
public static final int HW_SELF_CHECK_MASK_PICC      = 0x04;
public static final int HW_SELF_CHECK_MASK_FLASH    = 0x08;
public static final int HW_SELF_CHECK_MASK_KEY      = 0x10;
public static final int HW_SELF_CHECK_MASK_ALL      = 0xff;
```

Parameter	Type	Description
hwDeviceMask	int	The hardware part bit map. 0x01 - MSR 0x02 - ICCR 0x04 - CTLS 0x08 - Memory 0x10 - Keypad 0xFF - All hardware components.

Return	Type	Description
result	int	0 means success , others are failures

SysSetAppState

Set the App Run Status

```
void SysSetAppState(boolean inOrExit){...}
```

Parameter	Type	Description
inOrExit	boolean	<p>True - the system will mask the power long button to disable the screen, and disable the low power and other prompt boxes that will affect the Pin block processing.</p> <p>False - recovery</p> <p>Note: Applications use the SDK interface are recommended to be set it to true when the application is running, and must be restored when the application switch to the background. Otherwise, the system power key cannot be processed.</p>

SysGetAppState

Get App Run Status

```
boolean SysGetAppState(){...}
```

Return	Type	Description
state	boolean	<p>true -- set</p> <p>false -- not set</p>

SysSetWriteKeyResult

Set key filling result

```
void SysSetWriteKeyResult(int result){...}
```

Parameter	Type	Description
result	int	<p>0 - success</p> <p>1 - Unfilled</p> <p>2 - Filling failed</p> <p>3 - Filling is successful but verification failed</p>

SysGetWriteKeyResult

Get key filling result

```
int SysGetWriteKeyResult(){...}
```

Return	Type	Description
result	int	0 means success , others are failures Same as result of SysSetWriteKeyResult

PedSetKeyEntry

```
int PedSetKeyEntry(String keyName, byte[] keyData){...}
```

Parameter	Type	Description
keyName	String	
keyData	byte[]	

Return	Type	Description
result	int	0 means success , others are failures

PedGetKeyEntry

```
int PedGetKeyEntry(String keyName, PosByteArray keyData){...}
```

Parameter	Type	Description
keyName	String	
keyData	PosByteArray	

Return	Type	Description
result	int	0 means success , others are failures

SysUnlock

```
int sysunlock(){...}
```

Return	Type	Description
result	int	0 means success , others are failures

PedDeleteKeyEntry

```
int PedDeleteKeyEntry(String keyName){...}
```

Parameter	Type	Description
keyName	String	

Return	Type	Description
result	int	0 means success , others are failures

PedListKeyEntry

```
String[] PedListKeyEntry(String prefix){...}
```

Parameter	Type	Description
prefix	String	

Return	Type	Description
	String[]	

PedGetKeyEntrySize

```
long PedGetKeyEntrySize(String prefix){...}
```

Parameter	Type	Description
prefix	String	

Return	Type	Description
	long	

PedSetWriteKeyResult

```
void PedSetWriteKeyResult(int type, int result){...}
```

Parameter	Type	Description
type	int	
result	int	

PedGetWriteKeyResult

```
int PedGetWriteKeyResult(int type){...}
```

Parameter	Type	Description
type	int	

Return	Type	Description
result	int	

PedGetWriteKeyInfo

```
PedKeyInfo[] PedGetWriteKeyInfo(){...}
```

Return	Type	Description
keyInfo	PedKeyInfo []	

PedCalcDesEx

```
int PedCalcDesEx(int mode, byte[] keyBytes, byte[] dataBytes, byte[] ivBytes, PosByteArray rspBuf){...}
```

```
public static final int PED_CALC_DES_MODE_ECB_DEC      = 0x00;
public static final int PED_CALC_DES_MODE_ECB_ENC      = 0x01;
public static final int PED_CALC_DES_MODE_CBC_DEC      = 0x02;
public static final int PED_CALC_DES_MODE_CBC_ENC      = 0x03;
public static final int PED_CALC_DES_MODE_AES_ECB_DEC  = 0x10;
public static final int PED_CALC_DES_MODE_AES_ECB_ENC  = 0x11;
public static final int PED_CALC_DES_MODE_AES_CBC_DEC  = 0x12;
public static final int PED_CALC_DES_MODE_AES_CBC_ENC  = 0x13;
public static final int PED_CALC_DES_MODE_SM4_ECB_DEC  = 0x20;
public static final int PED_CALC_DES_MODE_SM4_ECB_ENC  = 0x21;
public static final int PED_CALC_DES_MODE_SM4_CBC_DEC  = 0x22;
public static final int PED_CALC_DES_MODE_SM4_CBC_ENC  = 0x23;
```

Parameter	Type	Description
mode	int	Same as mode of PedCalDes
keyBytes	byte[]	key
dataBytes	byte[]	Data to be encrypted/decryption
ivBytes	byte[]	
rspBuf	PosByteArray	Processed data

Return	Type	Description
result	int	0 means success , others are failures

registerListener

Register listener

```
void registerListener(EventListener listener){...}
```

```
hsmManage.registerListener(pinEventListener);
```

Parameter	Type	Description
listener	POIHsmManage.EventListener	

interface POIHsmManage.EventListener

```
public interface EventListener {
    public void onInfo(POIHsmManage secMgr, int what, int extra);

    public void onError(POIHsmManage secMgr, int what, int extra);

    public void onKeyboardShow(POIHsmManage secMgr, byte[] keys, int timeoutMs);

    public void onKeyboardInput(POIHsmManage secMgr, int numKeys);

    public void onPedPinBlockRet(POIHsmManage secMgr, int type, byte[] pinBuf);

    public void onPedVerifyPin(POIHsmManage secMgr, int type, byte[] rspBuf);

    public void onHwSelfCheckRet(POIHsmManage secMgr, int ret, int checkResult);

    public void onHwSensorTriggered(POIHsmManage secMgr, int triggered, byte[]
sensorValue, byte[] triggerTime);

    public void onPedKeyManagerRet(POIHsmManage secMgr, int ret);
}

class PinEventListener implements POIHsmManage.EventListener {
    /**
     * Payment\app\src\main\java\com\xc\payment\view\PasswordDialog.java
     * PinEventListener
     */
}

PinEventListener pinEventListener = new PinEventListener();
```

onInfo

Called to indicate an info or a warning

Parameter	Type	Description
secMgr	POIHsmManage	an instance of POIHsmManage
what	int	the type of info or warning
extra	int	an extra code, specific to the info. Typically implementation dependent

onError

Called to indicate an error

Parameter	Type	Description
secMgr	POIHsmManage	an instance of POIHsmManage
what	int	the type of error

Parameter	Type	Description
extra	int	an extra code, specific to the error. Typically implementation dependent

onKeyboardShow

Called when the Pin Block module should show UI with TP as Soft Keys

Parameter	Type	Description
secMgr	POIHsmManage	an instance of POIHsmManage
keys	byte[]	The keyboard sequence is displayed according to the Keys parameter. E.g 38 35 37 36 30 32 39 31 33 1B 34 0D (BCD)
timeoutMs	int	

onKeyboardInput

Button input value, input return value is defined as follows

Parameter	Type	Description
secMgr	POIHsmManage	an instance of POIHsmManage
numKeys	int	0xN -- returns 0xN keys correctly, max 12 keys 0x90 -- touch screen has been bounced by point 0x95 -- touch screen is not in range 0x96 -- Confirm button pressed 0x97 -- Cancel button pressed 0x98 -- number key pressed 0x99 -- The entered key has exceeded the set pin length

onPedPinBlockRet

PIN block returns, Support PED or PSAM.

Parameter	Type	Description
secMgr	POIHsmManage	an instance of POIHsmManage
type	int	0x83 -- DEFAULT PED_PINBLOCK_RET_TYPE_DEFAULT 0x92 -- PSAM PED_PINBLOCK_RET_TYPE_PSAM
pinBuf	type[]	08 62BC7A46801A6C46 0A FFFF9080102495000001 LEN + PINBLOCK + LEN + KSN

onPedVerifyPin

PIN verification, Support for Offline ciphertext PIN, offline plaintext PIN

Parameter	Type	Description
secMgr	POIHsmManage	an instance of POIHsmManage
type	int	0x86 -- CipherText PIN PED_VERIFY_PIN_TYPE_PLAIN 0x87 -- PlainText PIN PED_VERIFY_PIN_TYPE_CIPHER
rspBuf	int	Card status code. For example 9000, 6A83

onHwSelfCheckRet

Self-test results

Parameter	Type	Description
secMgr	POIHsmManage	an instance of POIHsmManage
type	int	Check type
checkResult	int	0x00 success

onHwSensorTriggered

Safe trigger

Parameter	Type	Description
secMgr	POIHsmManage	an instance of POIHsmManage
triggered	int	SP firmware status. 0x00 -- SP_STATUS_OK 0x01 -- SP_STATUS_TRIGGERED 0x02 -- SP_STATUS_LOCK 0x03 -- SP_STATUS_KEY_WIPED
sensorValue	byte[]	Trigger information
triggerTime	byte[]	Trigger Time

onPedKeyManageRet

Parameter	Type	Description
secMgr	POIHsmManage	an instance of POIHsmManage
ret	int	

unregisterListener

Unregister listener

```
void unregisterListener(EventListener listener){...}

hsmManage.unregisterListener(pinEventListener);
```

Parameter	Type	Description
listener	POIHsmManage.EventListener	

POICardManager

```
synchronized static POICardManager getDefault(Context context){...}

POICardManager cardManager = POICardManager.getDefault(mContext);
```

getSupportTypes

Types of card reading supported

```
int getSupportTypes(){...}

public static final int CARDREADER_TYPE_UNKNOWN = 0;
public static final int CARDREADER_TYPE_PSAM = (1 << 0);
public static final int CARDREADER_TYPE_ICC = (1 << 1);
public static final int CARDREADER_TYPE_MEMORY = (1 << 2);
public static final int CARDREADER_TYPE_PICC = (1 << 3);
public static final int CARDREADER_TYPE_MIFARE = (1 << 4);
public static final int CARDREADER_TYPE_VICC = (1 << 5);
public static final int CARDREADER_TYPE_SID = (1 << 6);
public static final int CARDREADER_TYPE_MAG = (1 << 7);
public static final int CARDREADER_TYPE_FELICA = (1 << 8);

int types = cardManager.getSupportTypes();
Log.d(TAG, "types = 0b" + Integer.toBinaryString(type));
// types = 0b1111111111111111
```

Return	Type	Description
types	int	Support ICC card: (types & POICardManager.CARDREADER_TYPE_ICC) > 0 The same for other types

getPsamCardReader

Get PosPsamCardReader object

```
synchronized PosPsamCardReader getPsamCardReader(){...}  
  
PosPsamCardReader psamCardReader = cardManager.getPsamCardReader();
```

Parameter	Type
psamCardReader	PosPsamCardReader

getIccCardReader

Get PosIccCardReader object

```
synchronized PosIccCardReader getIccCardReader(){...}  
  
PosIccCardReader iccCardReader = cardManager.getIccCardReader();
```

Parameter	Type
iccCardReader	PosIccCardReader

getMemoryCardReader

Get PosMemoryCardReader object

```
synchronized PosMemoryCardReader getMemoryCardReader(){...}  
  
PosMemoryCardReader memoryCardReader = cardManager.getMemoryCardReader();
```

Parameter	Type
memoryCardReader	PosMemoryCardReader

getPiccCardReader

Get PosPiccCardReader object

```
synchronized PosPiccCardReader getPiccCardReader(){...}  
  
PosPiccCardReader piccCardReader = cardManager.getPiccCardReader();
```

Parameter	Type
piccCardReader	PosPiccCardReader

getMifareCardReader

Get PosMifareCardReaderobject

```
synchronized PosMifareCardReader getMifareCardReader(){...}  
  
PosMifareCardReader mifareCardReader = cardManager.getMifareCardReader();
```

Parameter	Type
mifareCardReader	PosMifareCardReader

getViccCardReader

Get PosViccCardReaderobject

```
synchronized PosViccCardReader getViccCardReader(){...}  
  
PosViccCardReader viccCardReader = cardManager.getViccCardReader();
```

Parameter	Type
viccCardReader	PosViccCardReader

getSidCardReader

Get PosSidCardReader object

```
synchronized PosSidCardReader getSidCardReader(){...}  
  
PosSidCardReader sidCardReader = cardManager.getSidCardReader();
```

Parameter	Type
sidCardReader	PosSidCardReader

getMagCardReader

Get PosMagCardReader object

```
synchronized PosMagCardReader getMagCardReader(){...}  
  
PosMagCardReader magCardReader = cardManager.getMagCardReader();
```

Parameter	Type
magCardReader	PosMagCardReader

PosPsamCardReader

getNumberOfSlots

Get number of slots

```
int getNumberOfSlots(){...}

int num = psamCardReader.getNumberOfSlots();
Log.d(TAG, "slot number = " + num); // slot number = 1
```

Return	Type	Description
num	int	Number of slots

open

Operate PSAM card equipment function

```
int open(){...}
int open(int slot){...}
int open(int slot, int baudrate){...}
int open(int slot, int baudrate, int voltage){...}
int openWithPowerVol(int slot, int voltage){...}
int openWithWTX(int slot, int wtx){...}
int open(int slot, byte[] paramBuf){...}

public static final int DEFAULT_SLOT = 1;
```

Parameter	Type	Default	Description
slot	int	DEFAULT_SLOT slot of switchSlot	
baudrate	int	-/-	
voltage	int	-/-	
wtx	int	-/-	
paramBuf	byte[]	null	

Return	Type	Description
result	int	0 means success , others are failures

detect

Detecting the PASM card

```
int detect(){...}
```


Return	Type	Description
result	int	0 means success , others are failures

reset

Power-on and reset card

```
int reset(){...}
```

Return	Type	Description
result	int	0 means success , others are failures

getCardReaderInfo

Get card information list

```
PosCardReaderInfo getCardReaderInfo(){...}
```

Return	Type
cardInfo	PosCardReaderInfo

setTransmitApduTimeout

```
void setTransmitApduTimeout(long timeoutMs){...}
```

Parameter	Type	Description
timeoutMs	long	

setTransmitApduGetRspType

```
void setTransmitApduGetRspType(int type) {...}
```

Parameter	Type	Description
type	int	

transmitApu

APDU instruction interaction

```
int transmitApu(byte[] inBuf, PosByteArray rspBuf, PosByteArray swBuf){...}
```

Parameter	Type	Description
inBuf	byte[]	APDU command
rspBuf	PosByteArray	Card response result
swBuf	PosByteArray	Card response result

Return	Type	Description
result	int	0 means success , others are failures

close

Turn off card reader

```
int close(){...}
```

Return	Type	Description
result	int	0 means success , others are failures

switchSlot

```
int switchSlot(int slot, int baudrate){...}  
int switchSlot(int slot, int baudrate, int voltage){...}  
int switchSlotwithPowerVol(int slot, int voltage){...}  
int switchSlot(int slot, byte[] paramBuf){...}  
int setBaudrate(int baudrate){...}  
int setPowerVol(int voltage){...}
```

Parameter	Type	Description
slot	int	
baudrate	int	
voltage	int	

Parameter	Type	Description
paramBuf	byte[]	

Return	Type	Description
result	int	0 means success , others are failures

PoslccCardReader

open

Open ICC Card

```
int open(){...}
int open(int baudrate){...}
int open(int baudrate, int voltage){...}
int openWithPowerVol(int voltage){...}
int open(byte[] paramBuf){...}
```

Parameter	Type	Description
baudrate	int	
voltage	int	
paramBuf	byte[]	

Return	Type	Description
result	int	0 means success , others are failures

detect

Detecting the ICC card

```
int detect(){...}
```

Return	Type	Description
result	int	0 means success , others are failures

reset

Power-on and reset card

```
int reset(){...}
```

Return	Type	Description
result	int	0 means success , others are failures

getCardReaderInfo

Get card information list

```
PosCardReaderInfo getCardReaderInfo(){...}
```

Return	Type
cardInfo	PosCardReaderInfo

setTransmitApuTimeout

```
void setTransmitApuTimeout(long timeoutMs){...}
```

Parameter	Type	Description
timeoutMs	long	

setTransmitApuGetRspType

```
void setTransmitApuGetRspType(int type) {...}
```

Parameter	Type	Description
type	int	

transmitApu

APDU instruction interaction

```
int transmitApu(byte[] inBuf, PosByteArray rspBuf, PosByteArray swBuf){...}
```

Parameter	Type	Description
inBuf	byte[]	APDU command
rspBuf	PosByteArray	Card response result
swBuf	PosByteArray	Card response result

Return	Type	Description
result	int	0 means success , others are failures

close

Turn off card reader

```
int close(){...}
```

Return	Type	Description
result	int	0 means success , others are failures

setParameters

```
int setBaudrate(int baudrate){...}  
int setPowerVol(int voltage){...}  
int setParameters(byte[] paramBytes){...}
```

Parameter	Type	Description
baudrate	int	
voltage	int	
paramBuf	byte[]	

Return	Type	Description
result	int	0 means success , others are failures

PosMemoryCardReader

getSupportCardType

```
int getSupportCardType(){...}
```

Return	Type	Description
type	int	

isSupportCardType

```
boolean isSupportCardType(int cardType){...}
```

Parameter	Type	Description
cardType	int	

Return	Type	Description
isSupport	boolean	

setCardType

```
synchronized void setCardType(int cardType){...}
```

Parameter	Type	Description
cardType	int	

open

```
int open(){...}  
int open(PosByteArray rspBuf){...}  
int open(int cardType, PosByteArray rspBuf){...}
```

Parameter	Type	Description
cardType	int	
rspBuf	PosByteArray	

Return	Type	Description
result	int	0 means success , others are failures

close

```
int close(){...}
```

Return	Type	Description
result	int	0 means success , others are failures

read

```
int read(int address, int len, PosByteArray rspBuf){...}
int read(int zone, int address, int len, PosByteArray rspBuf){...}
int read(int zone, int address, int len, int opFlag, int specFlag, PosByteArray
rspBuf){...}
int read(int cardType, int zone, int address, int len, int opFlag, int specFlag,
PosByteArray rspBuf){...}
```

Parameter	Type	Default	Description
cardType	int		
zone	int		
address	int		
len	int		
opFlag	int		
specFlag	int		
rspBuf	PosByteArray		

Return	Type	Description
--------	------	-------------

Return	Type	Description
result	int	0 means success , others are failures

write

```
int write(int address, byte[] reqBuf){...}
int write(int zone, int address, byte[] reqBuf){...}
int write(int zone, int address, int opFlag, int specFlag, byte[] reqBuf){...}
int write(int cardType, int zone, int address, int opFlag, int specFlag, byte[] reqBuf){...}
```

Parameter	Type	Default	Description
cardType	int		
zone	int		
address	int		
opFlag	int		
specFlag	int		
reqBuf	byte[]		

Return	Type	Description
result	int	0 means success , others are failures

verify

```
int verify(byte[] reqBuf){...}
int verify(int zone, byte[] reqBuf){...}
int verify(int zone, int opFlag, int specFlag, byte[] reqBuf){...}
int verify(int cardType, int zone, int opFlag, int specFlag, byte[] reqBuf){...}
```

Parameter	Type	Default	Description
cardType	int		
zone	int		
opFlag	int		
specFlag	int		

Parameter	Type	Default	Description
reqBuf	byte[]		

Return	Type	Description
result	int	0 means success , others are failures

update

```
int update(byte[] reqBuf){...}
int update(int zone, byte[] reqBuf){...}
int update(int zone, int opFlag, int specFlag, byte[] reqBuf){...}
int update(int cardType, int zone, int opFlag, int specFlag, byte[] reqBuf){...}
```

Parameter	Type	Default	Description
cardType	int		
zone	int		
opFlag	int		
specFlag	int		
reqBuf	byte[]		

Return	Type	Description
result	int	0 means success , others are failures

pac

```
int pac(PosByteArray rspBuf){...}
int pac(int zone, PosByteArray rspBuf){...}
int pac(int zone, int opFlag, int specFlag, PosByteArray rspBuf){...}
int pac(int cardType, int zone, int opFlag, int specFlag, PosByteArray rspBuf)
{...}
```

Parameter	Type	Default	Description
cardType	int		
zone	int		

Parameter	Type	Default	Description
opFlag	int		
specFlag	int		
rspBuf	PosByteArray		

Return	Type	Description
result	int	0 means success , others are failures

status

```
int status(){...}
```

Return	Type	Description
status	int	

transmitRawCmd

```
int transmitRawCmd(byte[] reqBuf, PosByteArray rspBuf){...}
```

Parameter	Type	Description
reqBuf	byte[]	
rspBuf	PosByteArray	

Return	Type	Description
result	int	0 means success , others are failures

PosPiccCardReader

open

Open PICC Card

```
int open(){...}
```

Return	Type	Description
--------	------	-------------

Return	Type	Description
result	int	0 means success , others are failures

detect

Detecting the PICC card

```
int detect(){...}
int detect(String mode){...}

public static final int CARDREADER_DETECT_MODE_ISO14443 = 0x00;
public static final int CARDREADER_DETECT_MODE_EMV      = 0x01;
public static final int CARDREADER_DETECT_MODE_A        = 'A';
public static final int CARDREADER_DETECT_MODE_B        = 'B';
```

Parameter	Type	Value	Default	Description
mode	String	0x00 - MODE ISO14443 0x01 - MODE EMV 0x0A - MODE A 0x0B - MODE B	0x00	Picc card reader detect mode

Return	Type	Description
result	int	0 means success , others are failures

getCardReaderInfo

Get card information list

```
PosCardReaderInfo getCardReaderInfo(){...}
```

Return	Type
cardInfo	PosCardReaderInfo

transmitApu

APDU instruction interaction

```
int transmitApu(byte[] inBuf, PosByteArray rspBuf, PosByteArray swBuf){...}
```

Parameter	Type	Description
-----------	------	-------------

Parameter	Type	Description
inBuf	byte[]	APDU command
rspBuf	PosByteArray	Card response result
swBuf	PosByteArray	Card response result

Return	Type	Description
result	int	0 means success , others are failures

transmitRawCmd

```
int transmitRawCmd(byte[] reqBuf, PosByteArray rspBuf){...}
```

Parameter	Type	Description
reqBuf	byte[]	
rspBuf	PosByteArray	

Return	Type	Description
result	int	0 means success , others are failures

getCurrentOperation

```
int getCurrentOperation(){...}
```

Return	Type	Description
	int	

removeCard

Remove card reader

```
int removeCard(){...}
int removeCard(int mode){...}
int removeCard(int mode, int channel){...}
```

Parameter	Type	Description
mode	int	
channel	int	

Return	Type	Description
result	int	0 means success , others are failures

close

Turn off card reader

```
int close(){...}
```

Return	Type	Description
result	int	0 means success , others are failures

PosMifareCardReader

open

Open M1 Card

```
int open(){...}
int open(int cardType){...}
```

Parameter	Type	Description
cardType	int	

Return	Type	Description
result	int	0 means success , others are failures

detect

Detecting the M1 card

```
int detect(){...}
```

Return	Type	Description
result	int	0 means success , others are failures

getCardReaderInfo

Get card information list

```
PosCardReaderInfo getCardReaderInfo(){...}
```

Return	Type
cardInfo	PosCardReaderInfo

transmitApu

APDU instruction interaction

```
int transmitApu(byte[] inBuf, PosByteArray rspBuf){...}
```

Parameter	Type	Description
inBuf	byte[]	APDU command
rspBuf	PosByteArray	Card response result

Return	Type	Description
result	int	0 means success , others are failures

transmitRawCmd

```
int transmitRawCmd(byte[] reqBuf, PosByteArray rspBuf){...}
```

Parameter	Type	Description
reqBuf	byte[]	
rspBuf	PosByteArray	

Return	Type	Description
result	int	0 means success , others are failures

auth

M1 card authentication

```
int auth(int keyType, int blkNo, byte[] keyBuf){...}  
int auth(int keyType, int blkNo, byte[] keyBuf, byte[] serialNum){...}
```

Parameter	Type	Description
keyType	int	Key type: 'A' or 'a': A password 'B' or 'b': B password
blkNo	int	Block number
keyBuf	byte[]	Key data
serialNum	byte[]	Card serial number

Return	Type	Description
result	int	0 means success , others are failures

read

Read data from the specified block number of the M1 card

```
int read(int blkNo, PosByteArray rspBuf){...}  
int read(int blkNo, int len, PosByteArray rspBuf){...}  
int read(int index, int blkNo, int len, PosByteArray rspBuf){...}
```

Parameter	Type	Default	Description
index	int	0	
blkNo	int	-/-	Block number
len	int	16	
rspBuf	PosByteArray	-/-	Read data

Return	Type	Description
result	int	0 means success , others are failures

write

Write data to the specified block number of the M1 card

```
int write(int blkNo, byte[] buffer){...}  
int write(int index, int blkNo, byte[] buffer){...}
```

Parameter	Type	Description
index	int	index, default 0
blkNo	int	Block number
buffer	byte[]	Written data

Return	Type	Description
result	int	0 means success , others are failures

operate

```
int operate(int opType, int srcBlkNo, int dstBlkNo, int money){...}  
int operate(int opType, int srcBlkNo, int dstBlkNo, byte[] buffer){...}  
int operate(int opType, int index, int srcBlkNo, int dstBlkNo, byte[] buffer)  
{...}
```

Parameter	Type	Description
opType	int	
index	int	
srcBlkNo	int	
dstBlkNo	int	
money	int	
buffer	byte[]	

Return	Type	Description
result	int	0 means success , others are failures

removeCard

Remove card reader

```
int removeCard(){...}  
int removeCard(int mode){...}  
int removeCard(int mode, int channel){...}
```

Parameter	Type	Default	Description
mode	int	-1	
channel	int	-1	

Return	Type	Description
result	int	0 means success , others are failures

close

Turn off card reader

```
int close(){...}
```

Return	Type	Description
result	int	0 means success , others are failures

detectEx

```
int detectEx(int mode){...}  
int detectEx(String mode){...}
```

Parameter	Type	Description
mode	int String	mode Integer.toString(mode)

setCardType

```
int setCardType(int cardType){...}
```

Return	Type	Description
result	int	0 means success , others are failures

PosViccCardReader

open

Open VICC Card

```
int open(){...}
```

Return	Type	Description
result	int	0 means success , others are failures

inventory

Inventory from VICC Card

```
int inventory(PosByteArray rspBuf){...}
int inventory(int afi, PosByteArray rspBuf){...}
int inventory(int mode, int afi, PosByteArray rspBuf){...}

public static final int MODE_ISO15693 = ((0 << 4) | 0x05);
```

Parameter	Type	Default	Description
mode	int	MODE_ISO15693	
afi	int	-1	
rspBuf	PosByteArray	-/-	Card response result

Return	Type	Description
result	int	0 means success , others are failures

quiet

```
int quiet(byte[] uidBytes){...}
int quiet(int mode, byte[] uidBytes){...}
```

Parameter	Type	Default	Description
mode	int		
uidBytes	byte[]	-/-	

Return	Type	Description
result	int	0 means success , others are failures

readBlock

```
int readBlock(int blockNo, PosByteArray rspBuf){...}
int readBlock(byte[] uidBytes, int blockNo, PosByteArray rspBuf){...}
int readBlock(int blockNo, boolean rspSecStatus, PosByteArray rspBuf){...}
int readBlock(byte[] uidBytes, int blockNo, boolean rspSecStatus, PosByteArray
rspBuf){...}
int readBlock(int mode, byte[] uidBytes, int blockNo, boolean rspSecStatus,
PosByteArray rspBuf){...}
```

Parameter	Type	Default	Description
mode	int		
uidBytes	byte[]	null	
blockNo	int		
rspSecStaus	boolean	false	
rspBuf	PosByteArray		

Return	Type	Description
result	int	0 means success , others are failures

writeBlock

```
int writeBlock(int blockNo, byte[] reqBuf){...}
int writeBlock(byte[] uidBytes, int blockNo, byte[] reqBuf){...}
int writeBlock(int mode, byte[] uidBytes, int blockNo, byte[] reqBuf){...}
```

Parameter	Type	Description
mode	int	
uidBytes	byte[]	
blockNo	int	
reqBuf	byte[]	

Return	Type	Description
result	int	0 means success , others are failures

lockBlock

```
int lockBlock(int blockNo){...}
int lockBlock(byte[] uidBytes, int blockNo){...}
int lockBlock(int mode, byte[] uidBytes, int blockNo){...}
```

Parameter	Type	Description
mode	int	
uidBytes	byte[]	
blockNo	int	

Return	Type	Description
result	int	0 means success , others are failures

readMultiBlock

```

int readMultiBlock(int start, int NumOfBlk, PosByteArray rspBuf){...}
int readMultiBlock(int start, int NumOfBlk, boolean rspSecStatus, PosByteArray
rspBuf){...}
int readMultiBlock(byte[] uidBytes, int start, int NumOfBlk, boolean
rspSecStatus, PosByteArray rspBuf){...}
int readMultiBlock(int mode, byte[] uidBytes, int start, int NumOfBlk, boolean
rspSecStatus, PosByteArray rspBuf){...}

```

Parameter	Type	Default	Description
mode	int		
uidBytes	byte[]		
start	int		
NumOfBlk	int		
rspSecStatus	boolean		
rspBuf	PosByteArray	-/-	

Return	Type	Description
result	int	0 means success , others are failures

writeMultiBlock

```

int writeMultiBlock(int start, int NumOfBlk, byte[] reqBuf){...}
int writeMultiBlock(byte[] uidBytes, int start, int NumOfBlk, byte[] reqBuf)
{...}
int writeMultiBlock(int mode, byte[] uidBytes, int start, int NumOfBlk, byte[]
reqBuf){...}

```

Parameter	Type	Default	Description
mode	int		
iodBytes	byte[]		
start	int		
NumOfBlk	int		
reqBuf	byte[]		

Return	Type	Description
--------	------	-------------

Return	Type	Description
result	int	0 means success , others are failures

select

```
int select(byte[] uidBytes){...}
int select(int mode, byte[] uidBytes){...}
```

Parameter	Type	Description
mode	int	
uidBytes	byte[]	

Return	Type	Description
result	int	0 means success , others are failures

reset

Reset ViccCard

```
int reset(){...}
int reset(byte[] uidBytes){...}
int reset(int mode, byte[] uidBytes){...}
```

Parameter	Type	Description
mode	int	
uidBytes	byte[]	

Return	Type	Description
result	int	0 means success , others are failures

writeAFI

```
int writeAFI(int afi){...}
int writeAFI(byte[] uidBytes, int afi){...}
int writeAFI(int mode, byte[] uidBytes, int afi){...}
```

Parameter	Type	Description
mode	int	
uidBytes	byte[]	
afi	int	

Return	Type	Description
result	int	0 means success , others are failures

lockAFI

```
int lockAFI(){...}
int lockAFI(byte[] uidBytes){...}
int lockAFI(int mode, byte[] uidBytes){...}
```

Parameter	Type	Description
mode	int	
uidBytes	byte[]	

Return	Type	Description
result	int	0 means success , others are failures

writeDSFID

```
int writeDSFID(int dsfid){...}
int writeDSFID(byte[] uidBytes, int dsfid){...}
int writeDSFID(int mode, byte[] uidBytes, int dsfid){...}
```

Parameter	Type	Description
mode	int	
uidBytes	byte[]	
dsfid	int	

Return	Type	Description
result	int	0 means success , others are failures

lockDSFID

```
int lockDSFID(){...}  
int lockDSFID(byte[] uidBytes){...}  
int lockDSFID(int mode, byte[] uidBytes){...}
```

Parameter	Type	Description
mode	int	
uidBytes	byte[]	

Return	Type	Description
result	int	0 means success , others are failures

getSystemInfo

Get VICC Card info

```
int getSystemInfo(PosByteArray rspBuf){...}  
int getSystemInfo(byte[] uidBytes, PosByteArray rspBuf){...}  
int getSystemInfo(int mode, byte[] uidBytes, PosByteArray rspBuf){...}
```

Parameter	Type	Description
mode	int	
uidBytes	byte[]	
rspBuf	PosByteArray	

Return	Type	Description
result	int	0 means success , others are failures

getMultiBlockSecurityStatus


```

int getMultiBlockSecurityStatus(int start, int numOfBlock, PosByteArray rspBuf)
{...}
int getMultiBlockSecurityStatus(byte[] uidBytes, int start, int numOfBlock,
PosByteArray rspBuf){...}
int getMultiBlockSecurityStatus(int mode, byte[] uidBytes, int start, int
numOfBlock, PosByteArray rspBuf){...}

```

Parameter	Type	Description
mode	int	
uidBytes	byte[]	
start	int	
numOfBlock	int	
rspBuf	PosByteArray	

Return	Type	Description
result	int	0 means success , others are failures

transmitRawCmd

```

int transmitRawCmd(byte[] reqBuf, PosByteArray rspBuf){...}
int transmitRawCmd(int mode, byte[] reqBuf, PosByteArray rspBuf){...}

```

Parameter	Type	Description
mode	int	
reqBuf	byte[]	
rspBuf	PosByteArray	

Return	Type	Description
result	int	0 means success , others are failures

close

Turn off card reader

```

int close(){...}

```

Return	Type	Description
result	int	0 means success , others are failures

PosSidCardReader

open

Open Sid Card

```
int open(){...}
```

Return	Type	Description
result	int	0 means success , others are failures

detect

Detecting the Sid card

```
int detect(){...}
```

Return	Type	Description
result	int	0 means success , others are failures

getCardReaderInfo

Get card information list

```
PosCardReaderInfo getCardReaderInfo(){...}
```

Return	Type
cardInfo	PosCardReaderInfo

transmitCmd

CMD instruction interaction

```
int transmitCmd(byte[] reqBuf, PosByteArray rspBuf){...}
```

Parameter	Type	Description
-----------	------	-------------

Parameter	Type	Description
reqBuf	byte[]	
rspBuf	PosByteArray	

Return	Type	Description
result	int	0 means success , others are failures

transmitRawCmd

```
int transmitRawCmd(byte[] inBuf, PosByteArray rspBuf){...}
```

Parameter	Type	Description
inBuf	byte[]	
rspBuf	PosByteArray	

Return	Type	Description
result	int	0 means success , others are failures

removeCard

Remove card reader

```
int removeCard(){...}
int removeCard(int mode){...}
int removeCard(int mode, int channel){...}
```

Parameter	Type	Description
mode	int	
channel	int	

Return	Type	Description
result	int	0 means success , others are failures

close

Turn off card reader

```
int close(){...}
```

Return	Type	Description
result	int	0 means success , others are failures

setParameters

```
int setParameters(Parameters params){...}
```

Parameter	Type
params	Parameters

Return	Type	Description
result	int	0 means success , others are failures

getParameters

```
Parameters getParameters(){...}
```

Return	Type
params	Parameters

PosMagCardReader

open

Open Mag Card

```
int open(){
    return open(CARDREADER_DATA_TYPE_PLAIN, -1, -1, -1, (byte)'0', null);
}
int open(int keyType, int keyIndex, int mode, byte padding, byte[] initVector)
{...}
int open(int dataType, int keyType, int keyIndex, int mode, byte padding, byte[]
initVector){...}

public static final int CARDREADER_DATA_TYPE_PLAIN = 0x01;
```

```

public static final int CARDREADER_DATA_TYPE_ENCRYPT = 0x02;
public static final int CARDREADER_DATA_TYPE_ENCRYPT_ZIOSK = 0x03;
public static final int CARDREADER_DATA_TYPE_ENCRYPT_TRANSARMOR = 0x04;

public static final int CARDREADER_KEY_TYPE_TDK = 0x01;
public static final int CARDREADER_KEY_TYPE_DUKPT_MAC = 0x02;
public static final int CARDREADER_KEY_TYPE_DUKPT_DATA_REQUEST = 0x03;
public static final int CARDREADER_KEY_TYPE_DUKPT_DATA_RESPONSE = 0x04;
public static final int CARDREADER_KEY_TYPE_DUKPT_PIN = 0x05;

public static final int CARDREADER_MODE_ECB = 0x01;
public static final int CARDREADER_MODE_CBC = 0x02;

```

Parameter	Type	Description
dataType	int	Version type : 0x01 - PLAIN 0x02 - ENCRYPT 0x03 - ENCRYPT ZIOSK 0x04 - ENCRYPT TRANSARMOR
keyType	int	Version type : 0x01 - TDK 0x02 - DUKPT MAC 0x03 - DUKPT DATA REQUEST 0x04 - DUKPT DATA RESPONSE 0x05 - DUKPT PIN
keyIndex	int	key index
mode	int	Version type : 0x01 - ECB 0x02 - CBC
padding	byte	DES padding data
initVector	byte[]	Initial vector, CBC mode

Return	Type	Description
result	int	0 means success , others are failures

detect

Detecting the Mag card

```
int detect(){...}
```

Return	Type	Description
--------	------	-------------

Return	Type	Description
result	int	0 means success , others are failures

getCardReaderInfo

Get card information list

```
PosCardReaderInfo getCardReaderInfo(){...}
```

Return	Type
cardInfo	PosCardReaderInfo

getTraceData

Get trace data from MAG Card trace

```
byte[] getTraceData(int index){...}

public static final int CARDREADER_TRACE_INDEX_0 = 0x00;
public static final int CARDREADER_TRACE_INDEX_1 = 0x01;
public static final int CARDREADER_TRACE_INDEX_2 = 0x02;
public static final int CARDREADER_TRACE_INDEX_3 = 0x03;
```

Parameter	Type	Description
index	int	Trace index(Such as 1, 2, 3) CARDREADER_TRACE_INDEX_0 = 0x00 CARDREADER_TRACE_INDEX_1 = 0x01 CARDREADER_TRACE_INDEX_2 = 0x02 CARDREADER_TRACE_INDEX_3 = 0x03

Return	Type	Description
result	byte[]	Default: Trace Data,Failed to return null. Encryption: Ciphertext track Data, If encryption fails, the card number shielding data will be returned.

getTraceDataKsn

Obtain track data encryption SN from MAG card. Only used for DUKPT

```
byte[] getTraceDataKsn(int index){...}
```

Parameter	Type	Description
index	int	Trace index(Such as 1, 2, 3) Same as index of getTraceData

Return	Type	Description
result	byte[]	Ksn means success,Failed to return null

getTraceMaskData

Obtain the orbit data Mask card number from the MAG card. Only used in encryption mode

```
byte[] getTraceMaskData(int index){...}
```

Parameter	Type	Description
index	int	Trace index(Such as 1, 2, 3) Same as index of getTraceData

Return	Type	Description
result	byte[]	Mask Card means success,Failed to return null

close

Turn off card reader

```
int close(){...}
```

Return	Type	Description
result	int	0 means success , others are failures

class Parameters

```

public static class Parameters extends PosParameters {
    /**
     * Card reader reference attributes define.
     */
    public static final String KEY_TYPE = "sid-type";
    public static final String KEY_READER_RATE = "sid-reader-rate";
    public static final String KEY_CARD_RATE = "sid-card-rate";
    public static final String KEY_FRAME_MODE = "sid-frame-mode";
    public static final String KEY_FRAME_WAIT_TIME = "sid-frame-wait-time";
    public static final String KEY_FRAME_GUARD_TIME = "sid-frame-guard-time";
    public static final String KEY_FRAME_SIZE = "sid-frame-size";
    public static final String KEY_CRC_ENABLE = "sid-crc-enable";
}

```

Attribute	Description
KEY_TYPE	
KEY_READER_RATE	
KEY_CARD_RATE	
KEY_FRAME_MODE	
KEY_FRAME_WAIT_TIME	
KEY_FRAME_GUARD_TIME	
KEY_FRAME_SIZE	
KEY_CRC_ENABLE	

class PosCardReaderInfo

```

public class PosCardReaderInfo implements Parcelable {
    public int mCategory;
    public int mCardType;
    public int mCardChannel;
    public byte[] mSerialNum;
    public byte[] mAttribute;

    PosCardReaderInfo() {...}
    PosCardReaderInfo(int category, int cardType, int channel, byte[] sn, byte[]
attribute) {...}
    PosCardReaderInfo(PosCardReaderInfo o) {...}
}

public static final int CARDREADER_TYPE_UNKNOW = 0;
public static final int CARDREADER_TYPE_PSAM = (1 << 0);
public static final int CARDREADER_TYPE_ICC = (1 << 1);
public static final int CARDREADER_TYPE_MEMORY = (1 << 2);
public static final int CARDREADER_TYPE_PICC = (1 << 3);
public static final int CARDREADER_TYPE_MIFARE = (1 << 4);
public static final int CARDREADER_TYPE_VICC = (1 << 5);
public static final int CARDREADER_TYPE_SID = (1 << 6);

```



```
public static final int CARDREADER_TYPE_MAG      = (1 << 7);
public static final int CARDREADER_TYPE_FELICA  = (1 << 8);
```

Attribute	Type	Description
mCategory	int	Card Reader Category (ICC Card, PICC Card, Mag Card)
mCardType	int	Card type
mCardChannel	int	Card logical channel number
mSerialNum	byte[]	Serial number of the card, BCD encoding
mAttribute	byte[]	Card Attribute (ATR)

POIEmvCoreManager

```
POIEmvCoreManager emvCoreManager = POIEmvCoreManager.getDefault();
```

EmvSetAid

Add AID parameter

```
int EmvSetAid(PosEmvAid aid){...}
```

Parameter	Type
aid	PosEmvAid

Return	Type	Description
result	int	

EmvDeleteAid

Delete AID parameter

```
int EmvDeleteAid(){...}
```

Return	Type	Description
result	int	

EmvGetAid

Get AID parameter

```
List<PosEmvAid> EmvGetAid(){...}
```

Return	Type
param	List< PosEmvAid >

EmvSetCapk

Add CAPK parameter

```
int EmvSetCapk(PosEmvCapk capk){...}
```

Parameter	Type
capk	PosEmvCapk

Return	Type	Description
result	int	

EmvDeleteCapk

Delete CAPK parameter

```
int EmvDeleteCapk(){...}
```

Return	Type	Description
result	int	

EmvGetCapk

Get CAPK parameter

```
List<PosEmvCapk> EmvGetCapk(){...}
```

Return	Type
param	List< PosEmvCapk >

EmvSetExceptionFile

Add Exception File parameter

```
int EmvSetExceptionFile(PosEmvExceptionFile exceptionFile){...}
```

Parameter	Type
exceptionFile	PosEmvExceptionFile

Return	Type	Description
result	int	

EmvDeleteExceptionFile

Delete Exception File parameter

```
int EmvDeleteExceptionFile(){...}
```

Return	Type	Description
result	int	

EmvGetExceptionFile

Get Exception File parameter

```
List<PosEmvExceptionFile> EmvGetExceptionFile(){...}
```

Return	Type
param	List< PosEmvExceptionFile >

EmvSetRevocationIPK

Add CAPK Revocation parameter

```
int EmvSetRevocationIPK(PosEmvRevocationIPK revocationIPK){...}
```

Parameter	Type
revocationIPK	PosEmvRevocationIPK

Return	Type	Description
result	int	

EmvDeleteRevocationIPK

Delete CAPK Revocation parameter

```
int EmvDeleteRevocationIPK(){...}
```

Return	Type	Description
result	int	

EmvGetRevocationIPK

Get CAPK Revocation parameter

```
List<PosEmvRevocationIPK> EmvGetRevocationIPK(){...}
```

Return	Type
param	List< PosEmvRevocationIPK >

EmvSetTerminal

set terminal information in EMV kernel

```
int EmvSetTerminal(int type, Bundle bundle){...}
```

Parameter	Type	Description
type	int	TYPE_TERMINAL TYPE_INTERAC
bundle	android.os.Bundle	see the EmvTerminalConstraints class definition for details

Return	Type	Description
result	int	

EmvGetTerminal

get terminal information in EMV kernel

```
int EmvGetTerminal(int type, Bundle bundle){...}
```

Parameter	Type	Description
type	int	TYPE_TERMINAL TYPE_INTERAC
bundle	android.os.Bundle	see the EmvTerminalConstraints class definition for details

Return	Type	Description
result	int	

EmvSetDRL

Add DRL configuration parameter

```
int EmvSetDRL(int type, Bundle bundle){...}
```

Parameter	Type	Description
type	int	TYPE_VISA TYPE_AMEX
bundle	android.os.Bundle	see the EmvDrlConstraints class definition for details

Return	Type	Description
result	int	

EmvDeleteDRL

Delete DRL configuration parameter

```
int EmvDeleteDRL(int type){...}
```

Parameter	Type	Description
type	int	TYPE_VISA TYPE_AMEX

Return	Type	Description
result	int	

EmvGetDRL

Delete DRL configuration parameter

```
int EmvGetDRL(int type, Bundle bundle){...}
```

Parameter	Type	Description
type	int	TYPE_VISA TYPE_AAMEX
bundle	android.os.Bundle	see the EmvDrlConstraints class definition for details

Return	Type	Description
result	int	

EmvSetService

Add RuPay service parameter

```
int EmvSetService(Bundle bundle){...}
```

Parameter	Type	Description
bundle	android.os.Bundle	see the EmvServiceConstraints class definition for details

Return	Type	Description
result	int	

EmvDeleteService

Delete RuPay service parameter

```
int EmvDeleteService(Bundle bundle){...}
```

Parameter	Type	Description
bundle	android.os.Bundle	see the EmvServiceConstraints class definition for details

Return	Type	Description
result	int	

EmvGetService

Get RuPay service parameter

```
int EmvGetService(Bundle bundle){...}
```

Parameter	Type	Description
bundle	android.os.Bundle	see the EmvServiceConstraints class definition for details

Return	Type	Description
result	int	

AppleSetTerminal

```
int AppleSetTerminal(Bundle bundle){...}
```

Parameter	Type	Description
bundle	android.os.Bundle	see the AppleTerminalConstraints class definition for details

Return	Type	Description
result	int	

AppleGetTerminal

```
int AppleGetTerminal(Bundle bundle){...}
```

Parameter	Type	Description
bundle	android.os.Bundle	see the AppleTerminalConstraints class definition for details

Return	Type	Description
result	int	

AppleSetMerchant

```
int AppleSetMerchant(Bundle bundle){...}
```

Parameter	Type	Description
bundle	android.os.Bundle	

Return	Type	Description
result	int	

AppleDeleteMerchant

```
int AppleDeleteMerchant(){...}
```

Return	Type	Description
result	int	

AppleGetMerchant

```
int AppleGetMerchant(Bundle bundle){...}
```

Parameter	Type	Description
bundle	android.os.Bundle	

Return	Type	Description
result	int	

EmvGetChecksum

```
String EmvGetChecksum(int type){...}
```

Parameter	Type	Description
type	int	

Return	Type	Description
	String	

EmvGetVersion

```
String EmvGetVersion(int type){...}
```

Parameter	Type	Description
type	int	

Return	Type	Description
	String	

startTransaction

Start EMV process request

```
int startTransaction(Bundle data, IPosEmvCoreListener listener){...}
```

Parameter	Type	Description
bundle	android.os.Bundle	see the EmvTransDataConstraints class definition for details
listener	IPosEmvCoreListener	-/-

Return	Type	Description
result	int	see the PosEmvErrorCode class definition for details

interface IPosEmvCoreListener

```
interface IPosEmvCoreListener {  
  
    void onEmvProcess(int type, Bundle bundle);  
  
    void onSelectApplication(List<String> list, boolean isFirstSelect);  
  
    void onConfirmCardInfo(int mode, Bundle bundle);  
  
    void onKernelType(int type);  
  
    void onSecondTapCard();  
  
    void onRequestInputPin(Bundle bundle);  
  
    void onRequestOnlineProcess(Bundle bundle);  
  
    void onTransactionResult(int result, Bundle bundle);  
}
```

onEmvProcess

Check to the card. The transaction is ready to start

```
public static final int DEVICE_CONTACT      = 0x01;  
public static final int DEVICE_CONTACTLESS = 0x02;  
public static final int DEVICE_MAGSTRIPE   = 0x04;
```

Parameter	Type	Description
type	int	DEVICE_CONTACT DEVICE_CONTACTLESS DEVICE_MAGSTRIPE PosEmvErrorCode .EMV_MULTI_CONTACTLESS
bundle	android.os.Bundle	

onSelectApplication

When there are multiple candidate AIDs in an EMV transaction, the cardholder is required to make a selection. ICC only

Parameter	Type	Description
list	List	Candidate list name
isFirstSelect	boolean	First select

onConfirmCardInfo

Confirm card information. ICC only

```
public static final int CMD_TRY_OTHER_APPLICATION = 0x00;
public static final int CMD_AMOUNT_CONFIG        = 0x01;
public static final int CMD_ISSUER_REFERRAL      = 0x02;
```

Parameter	Type	Description
mode	int	Determine the type CMD_ISSUER_REFERRAL CMD_AMOUNT_CONFIG CMD_TRY_OTHER_APPLICATION
bundle	android.os.Bundle	see the EmvCardInfoConstraints class definition for details

onKernelType

Returns the type of card issuer. For example (VISA, MasterCard, AMEX, RuPay, etc.). PICC only

```
public static final int EMV_CARD_NOT           = 0x00;
public static final int EMV_CARD_VISA         = 0x01;
public static final int EMV_CARD_UNIONPAY     = 0x02;
public static final int EMV_CARD_MASTERCARD   = 0x03;
public static final int EMV_CARD_DISCOVER     = 0x04;
public static final int EMV_CARD_AMEX        = 0x05;
public static final int EMV_CARD_MIR         = 0x06;
public static final int EMV_CARD_RUPAY       = 0x07;
public static final int EMV_CARD_INTERAC     = 0x08;
```

Parameter	Type	Description
type	int	EMV_CARD_NOT EMV_CARD_VISA EMV_CARD_UNIONPAY EMV_CARD_MASTERCARD EMV_CARD_DISCOVER EMV_CARD_AMEX EMV_CARD_MIR EMV_CARD_RUPAY EMV_CARD_INTERAC

onSecondTapCard

Some issuers support script processing in PICC transactions. A second call is required. This callback is required by the application to display a prompt. Let the cardholder re-card. PICC only

onRequestInputPin

Input PIN. Support offline plain text PIN. Offline cipher text PIN. Online PIN

Parameter	Type	Description
bundle	android.os.Bundle	see the EmvPinConstraints class definition for details

onRequestOnlineProcess

Online connection is requested

Parameter	Type	Description
bundle	android.os.Bundle	see the EmvOnlineConstraints class definition for details

onTransactionResult

EMV process result

Parameter	Type	Description
result	int	see the PosEmvErrorCode class definition for details
bundle	android.os.Bundle	see the EmvResultConstraints class definition for details

stopTransaction

Stop EMV Transaction

```
int stopTransaction(){...}
```

Return	Type	Description
result	int	

onSetSelectResponse

When EMV processes request to select Application ([onSelectApplication](#)), EMV select the application

```
void onSetSelectResponse(int select){...}
```

Parameter	Type	Description
select	int	Application selection is from "1", "0" means cancellation

onSetCardInfoResponse

When EMV processes request to confirm the card information([onConfirmCardInfo](#)), it is called after user confirmation

```
void onSetCardInfoResponse(Bundle bundle){...}
```

Parameter	Type	Description
bundle	android.os.Bundle	

onSetPinResponse

When EMV processes request to PIN ([onRequestInputPin](#))

```
void onSetPinResponse(Bundle bundle){...}
```

Parameter	Type	Description
bundle	android.os.Bundle	see the EmvPinConstraints class definition for details

onSetOnlineResponse

When EMV processes request for on-line operation([onRequestOnlineProcess](#)) and the value is returned, the online data is passed into the process through this interface

```
void onSetOnlineResponse(Bundle bundle){...}
```

Parameter	Type	Description
bundle	android.os.Bundle	see the EmvOnlineConstraints class definition for details

class PosEmvAid

```
public class PosEmvAid implements Parcelable {  
    public byte[]  AID;  
    public byte[]  Version;  
    public boolean SelectIndicator;  
    public boolean TypeIndicator;  
    public byte[]  AcquirerIdentifier;  
    public byte[]  dDOL;  
    public byte[]  tDOL;  
    public byte[]  TACDenial;  
    public byte[]  TACOnline;  
}
```

```

    public byte[] TACDefault;
    public int Threshold;
    public int TargetPercentage;
    public int MaxTargetPercentage;
    public int FloorLimit;
    public int ContactlessTransLimit;
    public int ContactlessCVMLimit;
    public int ContactlessFloorLimit;
    public int DynamicTransLimit;
    public byte[] TerminalCountryCode;
    public byte[] MerchantCategoryCode;
    public byte[] TransCurrencyCode;
    public byte[] TransCurrencyExp;
    public byte[] TerminalType;
    public byte[] TerminalCapabilities;
    public byte[] AdditionalTerminalCapabilities;
    public byte[] TerminalRiskManagementData;
    public int CombinationType;
    public byte[] CombinationData;
}

```

Attribute	Type	Example	Description
AID	byte[]		Application id. 9F06
Version	byte[]		Application version. 9F09
SelectIndicator	boolean		Application select indicator: FULL_MATCH or PART_MATCH
TypeIndicator	boolean		Application type indicator: Contact or Contactless
AcquirerIdentifier	byte[]		Acquirer identifier
dDOL	byte[]		Default dDOL
tDOL	byte[]		Default tDOL
TACDenial	byte[]		TAC-denial
TACOnline	byte[]		TAC-online
TACDefault	byte[]		TAC-default
Threshold	int		Threshold of bias random selection
TargetPercentage	int		Target percentage of random selection
MaxTargetPercentage	int		The maximum target percentage of offset random selection
FloorLimit	int		Floor limit

Attribute	Type	Example	Description
ContactlessTransLimit	int		Contactless transaction limit
ContactlessCVMLimit	int		Contactless CVM required limits
ContactlessFloorLimit	int		Contactless Floor limit
DynamicTransLimit	int		Dynamic Trans limit
TerminalCountryCode	byte[]		Terminal Country Code 9F1A
MerchantCategoryCode	byte[]		Merchant Category Code 9F15
TransCurrencyCode	byte[]		Trans Currency Code 5F2A
TransCurrencyExp	byte[]		Trans Currency Exp 5F36
TerminalType	byte[]		Terminal Type
TerminalCapabilities	byte[]		Terminal capability
AdditionalTerminalCapabilities	byte[]		Additional Terminal Capabilities
TerminalRiskManagementData	byte[]		Terminal Risk Management Data
CombinationType	int		Combination Type
CombinationData	byte[]		Combination Data

class PosEmvCapk

```

public class PosEmvCapk implements Parcelable {

    public static final int ALGO_IND_RSA = 0x01;
    public static final int ALGO_IND_SM = 0x04;

    public static final int HASH_IND_NOT = 0x00;
    public static final int HASH_IND_SHA1 = 0x01;

    public byte[] RID;
    public byte CapkIndex;
    public byte[] Module;
    public byte[] Exponent;
    public byte[] Checksum;
    public byte AlgorithmInd;
    public byte HashInd;
}

```

Attribute	Type	Example	Description
RID	byte[]		Application registration service provider ID

Attribute	Type	Example	Description
CapkIndex	byte		Capk index
Module	byte[]		Module
Exponent	byte[]		Exponent
Checksum	byte[]		Checksum
AlgorithmInd	byte		Algorithm flag
HashInd	byte		HASH algorithm flag

class PosEmvExceptionFile

```
public class PosEmvExceptionFile implements Parcelable {

    public byte[] PAN;
    public byte[] SerialNo;
}
```

Attribute	Type	Example	Description
PAN	byte[]		Primary Account Number (PAN)
SerialNo	byte[]		serial number

class PosEmvRevocationIPK

```
public class PosEmvRevocationIPK implements Parcelable {

    public byte[] RID;
    public byte CapkIndex;
    public byte[] SerialNo;
}
```

Attribute	Type	Example	Description
RID	byte[]		RID
CapkIndex	byte		Capk index
SerialNo	byte[]		Serial number

class EmvTerminalConstraints

```
public static class EmvTerminalConstraints {

    public static final int TYPE_TERMINAL    = 1;
    public static final int TYPE_CONFIG      = 2;
    public static final int TYPE_VISA       = 3;
    public static final int TYPE_UNIONPAY   = 4;
}
```



```

    public static final int TYPE_MASTERCARD = 5;
    public static final int TYPE_DISCOVER = 6;
    public static final int TYPE_AMEX = 7;
    public static final int TYPE_MIR = 8;
    public static final int TYPE_RUPAY = 9;
    public static final int TYPE_INTERAC = 10;

    public static final String TERMINAL_ID = "TerminalId";
    public static final String TERMINAL_COUNTRY_CODE =
"TerminalCountryCode";
    public static final String TERMINAL_ENTRY_MODE = "TerminalEntryMode";
    public static final String MERCHANT_ID = "MerchantId";
    public static final String MERCHANT_CATEGORY_CODE =
"MerchantCategoryCode";
    public static final String MERCHANT_NAME = "MerchantName";
    public static final String TRANS_CURRENCY_CODE = "TransCurrencyCode";
    public static final String TRANS_CURRENCY_EXP = "TransCurrencyExp";
    public static final String TRANS_REFER_CURRENCY_CODE =
"TransReferCurrencyCode";
    public static final String TRANS_REFER_CURRENCY_EXP =
"TransReferCurrencyExp";
    public static final String IFD_SERIAL_NUMBER = "IfdSerialNumber";

    public static final String TERMINAL_TYPE = "TerminalType";
    public static final String TERMINAL_CAPABILITY = "TerminalCapability";
    public static final String TERMINAL_EX_CAPABILITY = "TerminalExCapability";

    public static final String PSE = "Pse";
    public static final String CARD_HOLDER_CONFIRM =
"CardHolderConfirm";
    public static final String LANGUAGE_SELECT = "LanguageSelect";
    public static final String DEFAULT_DDOL = "DefaultDDOL";
    public static final String DEFAULT_TDOL = "DefaultTDOL";
    public static final String BYPASS_PIN_ENTRY = "BypassPINEntry";
    public static final String SUBSEQUENT_BYPASS_PIN_ENTRY =
"SubsequentBypassPINEntry";
    public static final String GET_DATA_FOR_PIN_COUNTER =
"GetDataForPINCounter";
    public static final String FLOOR_LIMIT_CHECKING =
"FloorLimitChecking";
    public static final String RANDOM_TRANSACTION_SELECTION =
"RandomTransactionSelection";
    public static final String VELOCITY_CHECKING =
"VelocityChecking";
    public static final String EXCEPTION_FILE = "ExceptionFile";
    public static final String REVOCATION_ISSUER_PUBLIC_KEY =
"RevocationIssuerPublicKey";
    public static final String ISSUER_REFERRAL = "IssuerReferral";
    public static final String UNABLE_TO_GO_ONLINE =
"UnableToGoOnline";
    public static final String FORCED_ONLINE = "ForcedOnline";
    public static final String FORCED_ACCEPT = "ForcedAccept";

    public static final String CONFIG = "Config";

```

```

/**
 * 9F66 Terminal Transaction Qualifiers
 * Byte 1
 * bit 8: 1 = MSD supported (Mandatory no support)
 * bit 7: RFU (0)
 * bit 6: 1 = qVSDC supported
 * bit 5: 1 = EMV contact chip supported
 * bit 4: 1 = Offline-only reader
 * bit 3: 1 = Online PIN supported
 * bit 2: 1 = Signature supported
 * bit 1: 1 = Offline Data Authentication (ODA) for Online Authorizations
supported.
 * Byte 2
 * bit 8: 1 = Online cryptogram required
 * bit 7: 1 = CVM required
 * bits 6-1: RFU (00000)
 * Byte 3
 * bit 8: 1 = Issuer Update Processing supported
 * bit 7: 1 = Mobile functionality supported (Consumer Device CVM)
 * bits 6-1: RFU (000000)
 * Byte 4
 * RFU ('00')
 */
public static final String TAG_VISA_SET_QUALIFIERS = "9F66";
/**
 * DF1B Kernel Configuration
 * Byte 1
 * b1: Dr1SupportFlag
 */
public static final String TAG_VISA_SET_KERNEL_CONFIG = "DF1B";
/**
 * DF30 Entry Point
 * Byte 1
 * B8: StatusCheckSupportFlag
 * b7: ZeroAmountAllowedFlag
 * b6: ContactLessTransactionLimitFlag
 * b5: ContactLessFloorLimitFlag
 * b4: CVMRequiredLimitFlag
 */
public static final String TAG_VISA_SET_ENTRY_POINT = "DF30";
/**
 * DF32 Status Zero Amount Allowed Flag
 * Byte 1
 * 0x01: option1 = online cryptogram request
 * 0x02: option2 = not allowed
 */
public static final String TAG_VISA_SET_STATUS_ZERO_AMOUNT = "DF32";
/**
 * 9F66 Terminal Transaction Qualifiers
 * Byte 1
 * bit 8: RFU
 * bit 7: 1 = Full transaction flow in ContactLess interface Support
(Mandatory no support)
 * bit 6: 1 = EMV supported

```

```

    * bit 5: 1 = EMV contact chip supported
    * bit 4: 1 = Offline-only reader
    * bit 3: 1 = Online PIN supported
    * bit 2: 1 = Signature supported
    * bit 1: 1 = Offline Data Authentication (ODA) for Online Authorizations
supported.
    * Byte 2
    * bit 8: 1 = Online cryptogram required
    * bit 7: 1 = CVM required
    * bits 6-1: RFU (00000)
    * Byte 3
    * bit 8: RFU
    * bit 7: 1 = Mobile functionality supported (Consumer Device CVM)
    * bits 6-1: RFU (000000)
    * Byte 4
    * bit 8: 1 = fDDA v1.0 Supported (Mandatory support)
    */
public static final String TAG_UNIONPAY_SET_QUALIFIERS = "9F66";
/**
    * DF30 Entry Point
    * Byte 1
    * B8: StatusCheckSupportFlag
    * b7: ZeroAmountAllowedFlag
    * b6: ContactLessTransactionLimitFlag
    * b5: ContactLessFloorLimitFlag
    * b4: CVMRequiredLimitFlag
    */
public static final String TAG_UNIONPAY_SET_ENTRY_POINT = "DF30";
/**
    * DF32 Status Zero Amount Allowed Flag
    * Byte 1
    * 0x01: option1 = online cryptogram request
    * 0x02: option2 = not allowed
    */
public static final String TAG_UNIONPAY_SET_STATUS_ZERO_AMOUNT = "DF32";

/**
    * DF8118 CVM Capabilities CVM Required
    * Byte 1
    * B8: Plaintext PIN for ICC verification
    * b7: Enciphered PIN for online verification
    * b6: Signature (paper)
    * b5: Enciphered PIN for offline verification
    * b4: No CVM required
    */
public static final String TAG_MASTERCARD_SET_CVM_CAPABILITIES
= "DF8118";
/**
    * DF8119 CVM Capabilities No CVM Required
    * Byte 1
    * B8: Plaintext PIN for ICC verification
    * b7: Enciphered PIN for online verification
    * b6: Signature (paper)
    * b5: Enciphered PIN for offline verification
    * b4: No CVM required

```

```

    */
    public static final String TAG_MASTERCARD_SET_NO_CVM_CAPABILITIES
= "DF8119";
    /**
     * DF811E MagStripe CVM Capabilities CVM Required
     * Byte 1
     * B8..b5:
     * 0000: No CVM
     * 0001: Obtain Signature
     * 0010: Online PIN
     * 1111: N/A
     */
    public static final String TAG_MASTERCARD_SET_MAGSTRIPE_CVM_CAPABILITIES
= "DF811E";
    /**
     * DF812C MagStripe CVM Capabilities No CVM Required
     * Byte 1
     * B8..b5:
     * 0000: No CVM
     * 0001: Obtain Signature
     * 0010: Online PIN
     * 1111: N/A
     */
    public static final String TAG_MASTERCARD_SET_MAGSTRIPE_NO_CVM_CAPABILITIES
= "DF812C";
    /**
     * 9F6D MagStripe Application Version Number
     */
    public static final String TAG_MASTERCARD_SET_MAGSTRIPE_APP_VERSION
= "9F6D";
    /**
     * 9F7E Mobile Support Indicator
     */
    public static final String TAG_MASTERCARD_SET_MOBILE_SUPPORT_INDICATOR
= "9F7E";
    /**
     * DF810C Kernel ID
     */
    public static final String TAG_MASTERCARD_SET_KERNEL_ID
= "DF810C";
    /**
     * DF811A Default UDOL
     */
    public static final String TAG_MASTERCARD_SET_DEFAULT_UDOL
= "DF811A";
    /**
     * DF811B Kernel Configuration
     * Byte 1
     * b8: only EMV supported
     * b7: only MagStripe supported
     * b6: On device cardHolder verification supported
     * b5: Relay resistance protocol supported
     */
    public static final String TAG_MASTERCARD_SET_KERNEL_CONFIG
= "DF811B";

```

```

/**
 * DF8132 Minimum Relay Resistance Grace Period
 */
public static final String TAG_MASTERCARD_SET_RRP_MIN_GRACE
= "DF8132";
/**
 * DF8133 Maximum Relay Resistance Grace Period
 */
public static final String TAG_MASTERCARD_SET_RRP_MAX_GRACE
= "DF8133";
/**
 * DF8134 Terminal Expected Transmission Time For Relay Resistance C-APDU
 */
public static final String TAG_MASTERCARD_SET_RRP_CAPDU_EXPECTED
= "DF8134";
/**
 * DF8135 Terminal Expected Transmission Time For Relay Resistance R-APDU
 */
public static final String TAG_MASTERCARD_SET_RRP_RAPDU_EXPECTED
= "DF8135";
/**
 * DF8136 Relay Resistance Accuracy Threshold
 */
public static final String TAG_MASTERCARD_SET_RRP_ACCURACY_THRESHOLD
= "DF8136";
/**
 * DF8137 Relay Resistance Transmission Time Mismatch Threshold
 */
public static final String TAG_MASTERCARD_SET_RRP_MISMATCH_THRESHOLD
= "DF8137";

/**
 * 9F66 Terminal Transaction Qualifiers
 * Byte 1
 * bit 8: 1 = MSD supported
 * bit 7: RFU (0)
 * bit 6: 1 = EMV supported
 * bit 5: 1 = EMV contact chip supported
 * bit 4: 1 = Offline-only reader
 * bit 3: 1 = Online PIN supported
 * bit 2: 1 = Signature supported
 * bit 1: RFU
 * Byte 2
 * bit 8: 1 = Online cryptogram required
 * bit 7: 1 = CVM required
 * bits 6-1: RFU (00000)
 * Byte 3
 * bit 8: 1 = Issuer Update Processing supported
 * bit 7: 1 = Mobile functionality supported (Consumer Device CVM)
 * bits 6-1: RFU (000000)
 * Byte 4
 * RFU ('00')
 */
public static final String TAG_DISCOVER_SET_QUALIFIERS = "9F66";
/**

```

```

    * DF30 Entry Point
    * Byte 1
    * B8: StatusCheckSupportFlag
    * b7: ZeroAmountAllowedFlag
    * b6: ContactLessTransactionLimitFlag
    * b5: ContactLessFloorLimitFlag
    * b4: CVMRequiredLimitFlag
    */
public static final String TAG_DISCOVER_SET_ENTRY_POINT = "DF30";
/**
    * DF32 Status Zero Amount Allowed Flag
    * Byte 1
    * 0x01: option1 = online cryptogram request
    * 0x02: option2 = not allowed
    */
public static final String TAG_DISCOVER_SET_STATUS_ZERO_AMOUNT = "DF32";

/**
    * 9F6E Enhanced ContactLess Reader Capabilities
    * Byte 1
    * bit 8: 1 = EMV contact chip supported
    * bit 7: 1 = MSD supported
    * bit 6: 0 = (Fixed parameter)
    * bit 5: 1 = (Fixed parameter)
    * bit 4: 1 = Mobile supported
    * bit 3: 1 = Try Another Interface after a decline
    * bit 2: RFU
    * bit 1: RFU
    * Byte 2
    * bit 8: 1 = Mobile CVM supported
    * bit 7: 1 = Online PIN supported
    * bit 6: 1 = Signature supported
    * bits 5-1: RFU (00000)
    * Byte 3
    * bit 8: 1 = Terminal is offline only
    * bit 7: 1 = CVM Required
    * bits 6-1: RFU (000000)
    * Byte 4
    * bit 8: 1 = Exempt No CVM Checks
    * bit 7: 1 = Delayed Authorisations
    * RFU ('00')
    */
public static final String TAG_AMEX_SET_QUALIFIERS = "9F6E";
/**
    * DF1B Kernel Configuration
    * Byte 1
    * b1: Dr1SupportFlag
    */
public static final String TAG_AMEX_SET_KERNEL_CONFIG = "DF1B";
/**
    * DF30 Entry Point
    * Byte 1
    * B8: StatusCheckSupportFlag
    * b7: ZeroAmountAllowedFlag
    * b6: ContactLessTransactionLimitFlag

```

```

    * b5: ContactLessFloorLimitFlag
    * b4: CVMRequiredLimitFlag
    */
    public static final String TAG_AMEX_SET_ENTRY_POINT          = "DF30";
    /**
    * DF32 Status Zero Amount Allowed Flag
    * Byte 1
    * 0x01: option1 = online cryptogram request
    * 0x02: option2 = not allowed
    */
    public static final String TAG_AMEX_SET_STATUS_ZERO_AMOUNT = "DF32";

    /**
    * DF71 Transaction Processing Mode
    * Byte 1
    * bit 8: 1 = Online PIN supported
    * bit 7: 1 = Signature supported
    * bit 6: 1 = Mobile functionality supported (Consumer Device CVM)
    * bit 5: 1 = Terminal unable to go online
    * bit 4: 1 = EMV contact chip supported
    * bit 3: 1 = Offline-only reader
    * bit 2: 1 = Delayed Authorization
    * bit 1: 1 = Terminal is ATM
    * Byte 2
    * bit 8: 1 = Online cryptogram required
    * bit 7: 1 = CVM required
    * bit 6: 1 = Decline (AAC) cryptogram indicator
    * bit 5: 1 = Data Exchange was performed
    * bit 4: RFU (00)
    * bits 3-1: Kernel version
    */
    public static final String TAG_MIR_SET_QUALIFIERS          = "9F66";
    /**
    * DF30 Entry Point
    * Byte 1
    * B8: StatusCheckSupportFlag
    * b7: ZeroAmountAllowedFlag
    * b6: ContactLessTransactionLimitFlag
    * b5: ContactLessFloorLimitFlag
    * b4: CVMRequiredLimitFlag
    */
    public static final String TAG_MIR_SET_ENTRY_POINT          = "DF30";
    /**
    * DF32 Status Zero Amount Allowed Flag
    * Byte 1
    * 0x01: option1 = online cryptogram request
    * 0x02: option2 = not allowed
    */
    public static final String TAG_MIR_SET_STATUS_ZERO_AMOUNT = "DF32";
}

```

class EmvDrlConstraints

```
public static class EmvDrlConstraints {

    /**
     * DF01 variable DRL delimiter
     */
    public static final String TAG_DRL_SET_DELIMITER = "DF01";
    /**
     * 9F5A Application program Identifier
     */
    public static final String TAG_DRL_SET_PROGRAM_ID = "9F5A";
    /**
     * DF23 Reader ContactLess Transaction Limit
     */
    public static final String TAG_DRL_SET_TRANSACTION_LIMIT = "DF23";
    /**
     * DF24 Reader CVM Required Limit
     */
    public static final String TAG_DRL_SET_CVM_REQUIRED_LIMIT = "DF24";
    /**
     * DF25 Reader ContactLess Floor Limit
     */
    public static final String TAG_DRL_SET_FLOOR_LIMIT = "DF25";
    /**
     * DF30 Entry Point
     * Byte 1
     * B8: StatusCheckSupportFlag
     * b7: ZeroAmountAllowedFlag
     * b6: ContactLessTransactionLimitFlag
     * b5: ContactLessFloorLimitFlag
     * b4: CVMRequiredLimitFlag
     */
    public static final String TAG_DRL_SET_ENTRY_POINT = "DF30";
    /**
     * DF32 Status Zero Amount Allowed Flag
     * Byte 1
     * 0x01: option1 = online cryptogram request
     * 0x02: option2 = not allowed
     */
    public static final String TAG_DRL_SET_STATUS_ZERO_AMOUNT = "DF32";

    public static final int TYPE_VISA = 1;
    public static final int TYPE_AMEX = 2;

    public static final String CONFIG = "Config";
}
```


class EmvServiceConstraints

```
public static class EmvServiceConstraints {

    /**
     * DF01 variable Service delimiter
     */
    public static final String TAG_SERVICE_SET_DELIMITER = "DF01";

    /**
     * DF16 Service Identifier
     */
    public static final String TAG_SERVICE_SET_ID = "DF16";

    /**
     * DF17 Service Priority
     */
    public static final String TAG_SERVICE_SET_PRIORITY = "DF17";

    /**
     * DF18 Service Management
     */
    public static final String TAG_SERVICE_SET_MANAGEMENT = "DF18";

    /**
     * DF19 Service Data
     */
    public static final String TAG_SERVICE_SET_DATA = "DF19";

    /**
     * DF20 Service PRMiss
     */
    public static final String TAG_SERVICE_SET_PRMISS = "DF20";

    /**
     * DF21 Service PRMacq
     */
    public static final String TAG_SERVICE_SET_PRMACQ = "DF21";

    /**
     * DF02 variable PRMacq delimiter
     */
    public static final String TAG_PRMACQ_SET_DELIMITER = "DF02";

    /**
     * DF30 PRMacq index
     */
    public static final String TAG_PRMACQ_SET_INDEX = "DF30";

    /**
     * DF31 PRMacq Key
     */
    public static final String TAG_PRMACQ_SET_KEY = "DF31";

    /**
     * DF32 PRMacq Kcv
     */
    public static final String TAG_PRMACQ_SET_KCV = "DF32";

    public static final String CONFIG = "Config";
}
```

class AppleTerminalConstraints

```
public static class AppleTerminalConstraints {

    /**
     * Terminal P2 value for URL Only
     */
    public static final int PROTOCOL_URL_ONLY = 0x00;
    /**
     * Terminal P2 value for Full VAS
     */
    public static final int PROTOCOL_FULL_VAS = 0x01;
    /**
     * Terminal Capability Single Mode (VAS app OR payment)
     */
    public static final int CAPABILITY_SINGLE_MODE = 0x00;
    /**
     * Terminal Capability Dual Mode (VAS app AND payment)
     */
    public static final int CAPABILITY_DUAL_MODE = 0x01;
    /**
     * Terminal Capability VAS Only (no payment at all)
     */
    public static final int CAPABILITY_VAS_ONLY = 0x02;
    /**
     * Payment Only
     */
    public static final int CAPABILITY_PAYMENT_ONLY = 0x03;

    /**
     * DF01 variable Apple vas Merchant delimiter.
     */
    public static final String TAG_APPLE_SET_DELIMITER = "DF01";
    /**
     * 9F25 appleVas Merchant Id
     */
    public static final String TAG_APPLE_SET_MERCHANT_ID = "9F25";
    /**
     * 9F29 appleVas Merchant Url
     */
    public static final String TAG_APPLE_SET_MERCHANT_URL = "9F29";
    /**
     * 9F2B appleVas Filter
     */
    public static final String TAG_APPLE_SET_FILTER = "9F2B";

    public static final String PROTOCOL = "protocol";
    public static final String CAPABILITY = "capability";
    public static final String DATA = "data";
}
```

class EmvTransDataConstraints

```
public static class EmvTransDataConstraints {

    public static final int ENCRYPT_OPEN_CONTACT      = 0x01;
    public static final int ENCRYPT_OPEN_CONTACTLESS = 0x02;
    public static final int ENCRYPT_OPEN_MAGSTRIPE    = 0x04;

    public static final int ENCRYPT_KEY_MODE_TRANS_ARMOR = 0x01;

    public static final int ENCRYPT_TYPE_TDK          = 0x01;
    public static final int ENCRYPT_TYPE_DUKPT_MAC     = 0x02;
    public static final int ENCRYPT_TYPE_DUKPT_DATA_REQUEST = 0x03;
    public static final int ENCRYPT_TYPE_DUKPT_DATA_RESPONSE = 0x04;
    public static final int ENCRYPT_TYPE_DUKPT_PIN      = 0x05;
    public static final int ENCRYPT_TYPE_RSA           = 0x06;

    public static final int ENCRYPT_MODE_ECB = 0x01;
    public static final int ENCRYPT_MODE_CBC = 0x02;

    public static final String APPLE_VAS          = "appleVas";
    public static final String GOOGLE_SMART_TAP   = "googleSmartTap";
    public static final String TRANS_TYPE         = "transType";
    public static final String TRANS_AMOUNT       = "transAmount";
    public static final String TRANS_AMOUNT_OTHER = "transAmountOther";
    public static final String TRANS_DATE        = "transDate";
    public static final String TRANS_TIME        = "transTime";
    public static final String TRANS_MODE        = "transMode";
    public static final String TRANS_TIMEOUT     = "transTimeout";
    public static final String TRANS_FALLBACK    = "transFallback";
    public static final String AMOUNT_CONFIG     = "amountConfig";
    public static final String SPECIAL_CONTACT    = "specialContact";
    public static final String SPECIAL_MAGSTRIPE = "specialMagstripe";

    public static final String OPEN_ENCRYPT      = "openEncrypt";
    public static final String ENCRYPT_CONTACT   = "encryptContact";
    public static final String ENCRYPT_CONTACTLESS = "encryptContactless";
    public static final String ENCRYPT_MAGSTRIPE = "encryptMagstripe";
    public static final String ENCRYPT_KEY_MODE  = "encryptKeyMode";
    public static final String ENCRYPT_KEY_INDEX = "encryptKeyIndex";
    public static final String ENCRYPT_TYPE     = "encryptType";
    public static final String ENCRYPT_MODE     = "encryptMode";
    public static final String ENCRYPT_PADDING  = "encryptPadding";
    public static final String ENCRYPT_VECTOR   = "encryptVector";
    public static final String RSA_TRANS_ARMOR_POS_ID = "rsaTransArmorPosId";
    public static final String RSA_TRANS_ARMOR_KEY_ID = "rsaTransArmorKeyId";
    public static final String ENCRYPT_EMV_DATA  = "encryptEmvData";
    public static final String ENCRYPT_SHA1     = "encryptSHA1";
    public static final String ENCRYPT_BASE64    = "encryptBase64";

}
```

class EmvCardInfoConstraints

```
public static class EmvCardInfoConstraints {

    public static final String CARD      = "card";
    public static final String TRACK1    = "track1";
    public static final String TRACK2    = "track2";
    public static final String TRACK3    = "track3";

    public static final String OUT_CONFIRM      = "confirm";
    public static final String OUT_AMOUNT      = "amount";
    public static final String OUT_AMOUNT_OTHER = "amountOther";

}
```

class EmvPinConstraints

```
public static class EmvPinConstraints {

    public static final int VERIFY_SUCCESS      = 0;
    public static final int VERIFY_NO_PASSWORD = 1;
    public static final int VERIFY_PIN_BLOCK    = 2;
    public static final int VERIFY_ERROR        = 3;
    public static final int VERIFY_CANCELED     = 4;

    public static final String PIN_TYPE        = "pinType";
    public static final String PIN_ENCRYPT      = "pinEncrypt";
    public static final String PIN_CARD        = "pinCard";
    public static final String PIN_BYPASS      = "pinBypass";
    public static final String PIN_COUNTER     = "pinCounter";
    public static final String PIN_CARD_RANDOM = "pinCardRandom";
    public static final String PIN_MODULE      = "pinModule";
    public static final String PIN_EXPONENT    = "pinExponent";

    public static final String OUT_PIN_BLOCK      = "outPinBlock";
    public static final String OUT_PIN_TRY_COUNTER = "outPinTryCounter";
    public static final String OUT_PIN_VERIFY_RESULT = "outPinVerifyResult";

}
```

class EmvOnlineConstraints

```
public static class EmvOnlineConstraints {

    public static final int EMV_ONLINE_APPROVE      = 0;
    public static final int EMV_ONLINE_FAIL          = 1;
    public static final int EMV_ONLINE_DENIAL        = 2;
    public static final int EMV_ONLINE_REFER_TO_CARD_ISSUER = 3;

    public static final String EMV_DATA = "emvData";

    public static final String ENCRYPT_RESULT = "encryptResult";

}
```

```

    public static final String ENCRYPT_DATA = "encryptData";

    public static final String APPLE_RESULT = "appleResult";
    public static final String APPLE_DATA = "appleData";
    public static final String APPLE_MERCHANT = "appleMerchant";

    public static final String OUT_AUTH_RESP_CODE = "outAuthRespCode";
    public static final String OUT_AUTH_DATA = "outAuthData";
    public static final String OUT_AUTH_CODE = "outAuthCode";
    public static final String OUT_ISSUER_SCRIPT = "outIssuersScript";
}

```

class EmvResultConstraints

```

public static class EmvResultConstraints {

    public static final int CVM_NO_CVM = 0;
    public static final int CVM_SIGNATURE = 1;
    public static final int CVM_CONFIRMATION_CODE_VERIFIED = 2;

    public static final String EMV_DATA = "emvData";
    public static final String ENCRYPT_DATA = "encryptData";
    public static final String ENCRYPT_RESULT = "encryptResult";
    public static final String CVM = "cvm";
    public static final String SCRIPT_RESULT = "scriptResult";

    public static final String APPLE_RESULT = "appleResult";
    public static final String APPLE_DATA = "appleData";
    public static final String APPLE_MERCHANT = "appleMerchant";
}

```

class PosEmvErrorCode

```

public class PosEmvErrorCode {

    public final static int EXCEPTION_ERROR = 0xFF;
    public final static int PARAMETER_ERROR = 0xFE;

    public final static int EMV_OK = (0);
    public final static int EMV_APPROVED = (1);
    public final static int EMV_APPROVED_ONLINE = (2);
    public final static int EMV_DECLINED = (3);
    public final static int EMV_FORCE_APPROVED = (4);
    public final static int EMV_DELAYED_APPROVED = (5);
    public static final int APPLE_VAS_APPROVED = (6);

    public static final int EMV_CANCEL = (-1);
    public static final int EMV_TIMEOUT = (-2);
    public static final int EMV_COMMAND_FAIL = (-3);
    public static final int EMV_FALLBACK = (-4);
}

```

```

public static final int EMV_MULTI_CONTACTLESS    = (-5);
public static final int EMV_OTHER_ICC_INTERFACE = (-6);
public static final int EMV_APP_BLOCKED         = (-7);
public static final int EMV_CARD_BLOCKED        = (-8);
public static final int EMV_APP_EMPTY           = (-9);
public static final int EMV_NOT_ALLOWED          = (-10);
public static final int EMV_NOT_ACCEPTED        = (-11);
public static final int EMV_TERMINATED          = (-12);
public static final int EMV_SEE_PHONE           = (-13);
public static final int EMV_OTHER_INTERFACE     = (-14);
public static final int EMV_OTHER_ERROR         = (-999);

public static final int EMV_ENCRYPT_ERROR = (-30);
public static final int EMV_UNENCRYPTED   = (-31);

public static final int APPLE_VAS_UNTREATED      = (-40);
public static final int APPLE_VAS_FAILED         = (-41);
public static final int APPLE_VAS_WAITING_INTERVENTION = (-42);
public static final int APPLE_VAS_WAITING_ACTIVATION  = (-43);
}

```

Error code	Constant	Description
0xFE	EXCEPTION_ERROR	Exception error
0xFE	PARAMETER_ERROR	Parameter error
0	EMV_OK	Handle OK
1	EMV_APPROVED	Transaction approved
2	EMV_APPROVED_ONLINE	Transaction online approved
3	EMV_DECLINED	Transaction declined
4	EMV_FORCE_APPROVED	Transaction force approved
5	EMV_DELAYED_APPROVED	Transaction delayed approved
6	APPLE_VAS_APPROVED	Apple vas approval
-1	EMV_CANCEL	Trans cancel
-2	EMV_TIMEOUT	Trans Timeout
-3	EMV_COMMAND_FAIL	Read the card failure
-4	EMV_FALLBACK	Fallback
-5	EMV_MULTI_CONTACTLESS	Read multi contactless
-6	EMV_OTHER_ICC_INTERFACE	Not pure magnetic strip
-7	EMV_APP_BLOCKED	Application blocked
-8	EMV_CARD_BLOCKED	Card locked
-9	EMV_APP_EMPTY	No application

Error code	Constant	Description
-10	EMV_NOT_ALLOWED	Not allowed
-11	EMV_NOT_ACCEPTED	Not accepted
-12	EMV_TERMINATED	Terminated
-13	EMV_SEE_PHONE	See Phone
-14	EMV_OTHER_INTERFACE	Try another interface
-999	EMV_OTHER_ERROR	Other error exceptions
-30	EMV_ENCRYPT_ERROR	Trade encrypt error
-31	EMV_UNENCRYPTED	Trade unencrypted
-40	APPLE_VAS_UNTREATED	Apple vas untreated
-41	APPLE_VAS_FAILED	Apple vas failed
-42	APPLE_VAS_WAITING_INTERVENTION	Apple vas waiting intervention
-43	APPLE_VAS_WAITING_ACTIVATION	Apple vas waiting activation