ZCS POS Android Platform SDK Instructions and Examples

Version 1.0

SHENZHEN ZCS TECHNOLOGY CO., LTD

Overview

This document is applicable to the Android platform SDK of Smart POS (Z90, Z91, Z92, Z100), MPOS (Z70), all-in-one card reader module (Z45) and other products.

Environment

This document uses Android Studio as the development environment by default. If you use other development platforms to develop, please refer to the environment of the corresponding platform. This document is for reference only.

1. Import jar package

Copy the 'SmartPos_xxx.jar' file to the 'app\libs' directory. After the copy is complete, click the jar package, right-click—>add as library.

If you need to use the function of printing QR code, you also need to import the jar package of 'zxing', that is, copy the 'core-3.2.1.jar' file to the 'app\libs' directory. After the copy is complete, click the jar package and right-click—>add as library.

2. Import so library

Copy the 'armeabi-v7a' and 'arm64-v8a' directories to the 'src/main/jniLibs' directory.

The so library includes 'libSmartPosJni.so' and 'libEmvCoreJni.so'. Among them, 'libSmartPosJni.so' is the basic so, and 'libEmvCoreJni.so' is

the so related to the EMV function. If the EMV function is not required,

'libEmvCoreJni.so' can not be added.

Common class

1. Introduction

DriverManager　　Used to generate instance of each module operation class

Sys　　Used to obtain various device hardware information and system packaging interfaces

Printer　Used to print

CardReaderManager　　Used to find and operate various types of cards

EmvHandler　Used to run EMV

PinPadManager　Used to manage PinPad

Led　Used to operate LED

Beeper　Used to operate beeper

BluetoothHandler　Used for Z70 card reader

2. Get instance

Get each module operation class instance through various getXXX() functions of DriverManager

```
DriverManager mDriverManager = DriverManager.getInstance();
Sys mSys = mDriverManager.getBaseSysDevice();
CardReaderManager mCardReadManager = mDriverManager.getCardReadManager();
EmvHandler mEmvHandler = EmvHandler.getInstance();
```

```
PinPadManager mPadManager = mDriverManager.getPadManager();

Printer mPrinter = mDriverManager.getPrinter();

Beeper mBeeper = mDriverManager.getBeeper();

Led mLed = mDriverManager.getLedDriver();

BluetoothHandler mBluetoothHandler = mDriverManager.getBluetoothHandler();
```

initialization

All interfaces need to initialize the SDK before use.

1. Default (for Z90, Z91, Z92, Z100)

Refer to the following code.

```java
private void initSdk() {
    int status = mSys.sdkInit();
    if(status != SdkResult.SDK_OK) {
        mSys.sysPowerOn();
        try {
            Thread.sleep(1000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
    status = mSys.sdkInit();
    if(status != SdkResult.SDK_OK) {
        //init failed.
    }
}
```

2. Bluetooth (for Z70)

Refer to the following code.

```java
private void initSdk() {
    // Config the SDK base info
    mSys = mDriverManager.getBaseSysDevice();
    mSys.showLog(true);
    mBluetoothManager = BluetoothManager.getInstance()
    .setContext(mActivity)
```

```java
    .setBluetoothListener(new BluetoothListener() {

        @Override
        public boolean isReader(BluetoothDevice bluetoothDevice) {
            // Get device searched by bluetooth
            mAdapter.addDevice(bluetoothDevice);
            mAdapter.notifyDataSetChanged();
            return false;
        }
        @Override
        public void startedConnect(BluetoothDevice device) {
            Log.e(TAG, "startedConnect: ");
        }
        @Override
        public void connected(BluetoothDevice device) {
            Log.e(TAG, "connected: ");
            mHandler.obtainMessage(MSG_TOAST, "Connected").sendToTarget();
            int sdkInit = mSys.sdkInit(ConnectTypeEnum.BLUETOOTH);
            String initRes = (sdkInit == SdkResult.SDK_OK) ? getString(R.string.init_success
) : SDK_Result.obtainMsg(mActivity, sdkInit);
            // mBluetoothManager.connect called in sub thread, u need to switch to main thre
ad when u need to change ui
            mHandler.obtainMessage(MSG_TOAST, initRes).sendToTarget();
        }
        @Override
        public void disConnect() {
            Log.e(TAG, "disConnect: ");
            mHandler.obtainMessage(MSG_TOAST, "Disconnect").sendToTarget();
        }
        @Override
        public void startedDiscovery() {
            Log.e(TAG, "startedDiscovery: ");
        }
        @Override
        public void finishedDiscovery() {
            Log.e(TAG, "finishedDiscovery: ");
        }
    })
    .init();
}
```

3. USB (for Z45)

Refer to the following code.

```java
private int openUsb() {
    if (mUsbHandler != null) {
        mUsbHandler.close();
    }
    mUsbHandler = UsbHandler.getInstance().setContext(this).init();
    int nRet = mUsbHandler.connect();
    if (nRet == USBConstants.USB_NO_PERMISSION) {
        mUsbHandler.checkPermission();
        nRet = mUsbHandler.connect();
    } else if (nRet == USBConstants.USB_NO_USB_DEVICE || nRet == USBConstants.USB_NOT_FIND_D
EVICE) {
        SystemClock.sleep(2000);
        if (mUsbHandler != null) {
            mUsbHandler.close();
        }
        mUsbHandler = UsbHandler.getInstance().setContext(this).init();
        nRet = mUsbHandler.connect();
    }
    showLog("openUsb: " + nRet);
    if (nRet == 0) {
        nRet = mSys.sdkInit(ConnectTypeEnum.USB);
        showLog("sdkInit:" + nRet);
    }
    return nRet;
}
```

Print

This chapter will show some methods of printing functions, please adjust the specific usage according to actual needs. The SDK needs to be initialized before printing, please refer to the previous chapter.

1. Print text

```java
private void printText() {
    int printStatus = mPrinter.getPrinterStatus();
    if (printStatus == SdkResult.SDK_PRN_STATUS_PAPEROUT) {
        //out of paper
    } else {
```

```
        PrnStrFormat format = new PrnStrFormat();

        format.setTextSize(30);

        format.setAli(Layout.Alignment.ALIGN_CENTER);

        format.setStyle(PrnTextStyle.BOLD);

        format.setFont(PrnTextFont.CUSTOM);

        format.setPath(Environment.getExternalStorageDirectory() + "/fonts/simsun.ttf");

        mPrinter.setPrintAppendString("POS SALES SLIP", format);

        format.setTextSize(25);

        format.setStyle(PrnTextStyle.NORMAL);

        format.setAli(Layout.Alignment.ALIGN_NORMAL);

        mPrinter.setPrintAppendString(" ", format);

        mPrinter.setPrintAppendString("MERCHANGT NAME:" + " Test ", format);

        mPrinter.setPrintAppendString("MERCHANT NO:" + " 123456789012345 ", format);

        mPrinter.setPrintAppendString("TERMINAL NAME:" + " 12345678 ", format);

        mPrinter.setPrintAppendString("OPERATOR NO:" + " 01 ", format);

        mPrinter.setPrintAppendString("CARD NO: ", format);

        format.setAli(Layout.Alignment.ALIGN_CENTER);

        format.setTextSize(30);

        format.setStyle(PrnTextStyle.BOLD);

        mPrinter.setPrintAppendString("6214 44** **** **** 7816", format);

        format.setAli(Layout.Alignment.ALIGN_NORMAL);

        format.setStyle(PrnTextStyle.NORMAL);

        format.setTextSize(25);mPrinter.setPrintAppendString(" ----------------------------
", format);

        mPrinter.setPrintAppendString(" ", format);

        mPrinter.setPrintAppendString(" ", format);

        mPrinter.setPrintAppendString(" ", format);

        mPrinter.setPrintAppendString(" ", format);

        printStatus = mPrinter.setPrintStart();

    }

}
```

2. Print QR code (the jar package of zxing needs to be imported)

```
private void printQrcode(String qrString) {

    int printStatus = mPrinter.getPrinterStatus();

    if (printStatus != SdkResult.SDK_PRN_STATUS_PAPEROUT) {

        mPrinter.setPrintAppendQRCode(qrString, 200, 200, Layout.Alignment.ALIGN_CENTER);

        printStatus = mPrinter.setPrintStart();

    }

}
```

3. Print barcodes (the jar package of zxing needs to be imported)

This section only shows barcodes whose printing format is CODE_128, other formats need to be replaced with corresponding parameters. Must be in a format supported by zxing.

```java
private void printBarCode128(String barcodeString) {
    int printStatus = mPrinter.getPrinterStatus();
    if (printStatus != SdkResult.SDK_PRN_STATUS_PAPEROUT) {
        mPrinter.setPrintAppendBarCode(getActivity(), barcodeString, 360, 100, true, Layout.
Alignment.ALIGN_CENTER, BarcodeFormat.CODE_128);
        printStatus = mPrinter.setPrintStart();
    }
}
```

4. Print bitmap

```java
private void printBitmap(Bitmap bitmap) {
    int printStatus = mPrinter.getPrinterStatus();
    if (printStatus != SdkResult.SDK_PRN_STATUS_PAPEROUT) {
        mPrinter.setPrintAppendBitmap(bitmap, Layout.Alignment.ALIGN_CENTER);
        printStatus = mPrinter.setPrintStart();
    }
}
```

5. Print label

Printing label requires hardware support, otherwise it will get an error.

```java
private void printLabel(Bitmap bitmap) {
    int printStatus = mPrinter.getPrinterStatus();
    if (printStatus != SdkResult.SDK_PRN_STATUS_PAPEROUT) {
        mPrinter.printLabel(bitmap);
    }
}
```

Card

1. some common methods

This section lists some common methods that will be used to read the card. The sdk needs to be initialized before reading the card, please refer to the previous chapter.

```java
    private DriverManager mDriverManager = DriverManager.getInstance();
    private CardReaderManager mCardReadManager = mDriverManager.getCardReadManager();
    private static final int READ_TIMEOUT = 60 * 1000;
    private ProgressDialog mProgressDialog;


    private void showSearchCardDialog(@StringRes int title, @StringRes int msg) {
        mProgressDialog = (ProgressDialog) DialogUtils.showProgress(getActivity(), getString
(title), getString(msg), new DialogInterface.OnCancelListener() {
            @Override
            public void onCancel(DialogInterface dialog) {
                mCardReadManager.cancelSearchCard();
            }
        });
    }


    private static String cardInfoToString(CardInfoEntity cardInfoEntity) {
        if (cardInfoEntity == null)
            return null;
        StringBuilder sb = new StringBuilder();
        sb.append("Resultcode:\t" + cardInfoEntity.getResultcode() + "\n")
                .append(cardInfoEntity.getCardExistslot() == null ? "" : "Card type:\t" + ca
rdInfoEntity.getCardExistslot().name() + "\n")
                .append(cardInfoEntity.getCardNo() == null ? "" : "Card no:\t" + cardInfoEnt
ity.getCardNo() + "\n")
                .append(cardInfoEntity.getRfCardType() == 0 ? "" : "Rf card type:\t" + cardI
nfoEntity.getRfCardType() + "\n")
                .append(cardInfoEntity.getRFuid() == null ? "" : "RFUid:\t" + new String(car
dInfoEntity.getRFuid()) + "\n")
                .append(cardInfoEntity.getAtr() == null ? "" : "Atr:\t" + cardInfoEntity.get
Atr() + "\n")
                .append(cardInfoEntity.getTk1() == null ? "" : "Track1:\t" + cardInfoEntity.
```

```
getTk1() + "\n")
                .append(cardInfoEntity.getTk2() == null ? "" : "Track2:\t" + cardInfoEntity.
getTk2() + "\n")
                .append(cardInfoEntity.getTk3() == null ? "" : "Track3:\t" + cardInfoEntity.
getTk3() + "\n")
                .append(cardInfoEntity.getExpiredDate() == null ? "" : "expiredDate:\t" + ca
rdInfoEntity.getExpiredDate() + "\n")
                .append(cardInfoEntity.getServiceCode() == null ? "" : "serviceCode:\t" + ca
rdInfoEntity.getServiceCode());
        return sb.toString();
    }


    private String rfCardTypeToString(byte rfCardType) {
        String type = "";
        switch (rfCardType) {
            case SdkData.RF_TYPE_A:
                type = "RF_TYPE_A";
                break;
            case SdkData.RF_TYPE_B:
                type = "RF_TYPE_B";
                break;
            case SdkData.RF_TYPE_MEMORY_A:
                type = "RF_TYPE_MEMORY_A";
                break;
            case SdkData.RF_TYPE_FELICA:
                type = "RF_TYPE_FELICA";
                break;
            case SdkData.RF_TYPE_MEMORY_B:
                type = "RF_TYPE_MEMORY_B";
                break;
        }
        return type;
    }
```

2. IC Card

This section will show how to read the information of IC card.

```
    private void searchICCard() {
        showSearchCardDialog(R.string.waiting, R.string.msg_ic_card);
        mCardReadManager.cancelSearchCard();
        mCardReadManager.searchCard(CardReaderTypeEnum.IC_CARD, READ_TIMEOUT, mICCardSearchC
ardListener);
```

```java
        }

    private OnSearchCardListener mICCardSearchCardListener = new OnSearchCardListener() {
        @Override
        public void onCardInfo(CardInfoEntity cardInfoEntity) {
            mProgressDialog.dismiss();
            readICCard();
        }

        @Override
        public void onError(int i) {
            mProgressDialog.dismiss();
            showReadICCardErrorDialog(i);
        }

        @Override
        public void onNoCard(CardReaderTypeEnum cardReaderTypeEnum, boolean b) {

        }
    };

    public static final byte[] APDU_SEND_IC = {0x00, (byte) 0xA4, 0x04, 0x00, 0x0E, 0x31, 0x
50, 0x41, 0x59, 0x2E, 0x53, 0x59, 0x53, 0x2E, 0x44, 0x44, 0x46, 0x30, 0x31, 0X00};
    private void readICCard() {
        ICCard icCard = mCardReadManager.getICCard();
        int result = icCard.icCardReset(CardSlotNoEnum.SDK_ICC_USERCARD);
        if (result == SdkResult.SDK_OK) {
            int[] recvLen = new int[1];
            byte[] recvData = new byte[300];
            result = icCard.icExchangeAPDU(CardSlotNoEnum.SDK_ICC_USERCARD, APDU_SEND_IC, re
cvData, recvLen);
            if (result == SdkResult.SDK_OK) {
                final String apduRecv = StringUtils.convertBytesToHex(recvData).substring(0,
recvLen[0] * 2);
                CardFragment.this.getActivity().runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                        DialogUtils.show(getActivity(), "Read IC card result", apduRecv);
                    }
                });
            } else {
                showReadICCardErrorDialog(result);
            }
        } else {
```

```
            showReadICCardErrorDialog(result);
        }
        icCard.icCardPowerDown(CardSlotNoEnum.SDK_ICC_USERCARD);
    }


    private void showReadICCardErrorDialog(final int errorCode) {
        CardFragment.this.getActivity().runOnUiThread(new Runnable() {
            @Override
            public void run() {
                DialogUtils.show(getActivity(), "Read IC card failed", "Error code = " + err
orCode);
            }
        });
    }
```

3. PSAM Card

This section will show how to read a PSAM card. The code only shows
how to obtain the PSAM card in slot 1. If you want to read the PSAM card
in other card slots, you need to replace the corresponding parameters.

```
    private void searchPSAM1() {
        showSearchCardDialog(R.string.waiting, R.string.psam_ic_card);
        mCardReadManager.cancelSearchCard();
        mCardReadManager.searchCard(CardReaderTypeEnum.PSIM1, READ_TIMEOUT, mPSAM1SearchCard
Listener);
    }


    private OnSearchCardListener mPSAM1SearchCardListener = new OnSearchCardListener() {
        @Override
        public void onCardInfo(CardInfoEntity cardInfoEntity) {
            mProgressDialog.dismiss();
            readPSAM1();
        }

        @Override
        public void onError(int i) {
            mProgressDialog.dismiss();
            showReadPSAM1ErrorDialog(i);
        }
```

```java
        @Override
        public void onNoCard(CardReaderTypeEnum cardReaderTypeEnum, boolean b) {


        }
    };


    public static final byte[] APDU_SEND_RANDOM = {0x00, (byte) 0x84, 0x00, 0x00, 0x08};
    private void readPSAM1() {
        ICCard icCard = mCardReadManager.getICCard();
        int result = icCard.icCardReset(CardSlotNoEnum.SDK_ICC_SAM1);
        if (result == SdkResult.SDK_OK) {
            int[] recvLen = new int[1];
            byte[] recvData = new byte[300];
            result = icCard.icExchangeAPDU(CardSlotNoEnum.SDK_ICC_SAM1, APDU_SEND_RANDOM, re
cvData, recvLen);
            if (result == SdkResult.SDK_OK) {
                final String apduRecv = StringUtils.convertBytesToHex(recvData).substring(0,
 recvLen[0] * 2);
                CardFragment.this.getActivity().runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                        DialogUtils.show(getActivity(), "Read PSAM1 result", apduRecv);
                    }
                });
            } else {
                showReadPSAM1ErrorDialog(result);
            }
        } else {
            showReadPSAM1ErrorDialog(result);
        }
        icCard.icCardPowerDown(CardSlotNoEnum.SDK_ICC_SAM1);
    }

    private void showReadPSAM1ErrorDialog(final int errorCode) {
        CardFragment.this.getActivity().runOnUiThread(new Runnable() {
            @Override
            public void run() {
                DialogUtils.show(getActivity(), "Read PSAM1 failed", "Error code = " + error
Code);
            }
        });
    }
```

4. Magnetic stripe card

This section will show how to read magnetic stripe card information.

```java
    private void searchMagnetCard() {
        showSearchCardDialog(R.string.waiting, R.string.msg_magnet_card);
        mCardReadManager.cancelSearchCard();
        mCardReadManager.searchCard(CardReaderTypeEnum.MAG_CARD, READ_TIMEOUT, mMagnetCardSe
archCardListener);
    }


    private OnSearchCardListener mMagnetCardSearchCardListener = new OnSearchCardListener()
{
        @Override
        public void onCardInfo(CardInfoEntity cardInfoEntity) {
            mProgressDialog.dismiss();
            readMagnetCard();
        }

        @Override
        public void onError(int i) {
            mProgressDialog.dismiss();
            showReadMagnetCardErrorDialog(i);
        }

        @Override
        public void onNoCard(CardReaderTypeEnum cardReaderTypeEnum, boolean b) {

        }
    };
    private void readMagnetCard() {
        MagCard magCard = mCardReadManager.getMAGCard();
        final CardInfoEntity cardInfoEntity = magCard.getMagReadData();
        if (cardInfoEntity.getResultcode() == SdkResult.SDK_OK) {
            CardFragment.this.getActivity().runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    DialogUtils.show(getActivity(), "Read Magnet card result", cardInfoToStr
ing(cardInfoEntity));
                }
            });
        } else {
            showReadMagnetCardErrorDialog(cardInfoEntity.getResultcode());
        }
```

```
        magCard.magCardClose();
    }


    private void showReadMagnetCardErrorDialog(final int errorCode) {
        CardFragment.this.getActivity().runOnUiThread(new Runnable() {
            @Override
            public void run() {
                DialogUtils.show(getActivity(), "Read magnetic card failed", "Error code = "
+ errorCode);
            }
        });
    }
```

5. Contactless card

This section will show how to read contactless card information.

```
    private void searchRfCard() {
        showSearchCardDialog(R.string.waiting, R.string.msg_contactless_card);
        mCardReadManager.cancelSearchCard();
        mCardReadManager.searchCard(CardReaderTypeEnum.RF_CARD, READ_TIMEOUT, mRfCardSearchC
ardListener);
    }


    private OnSearchCardListener mRfCardSearchCardListener = new OnSearchCardListener() {
        @Override
        public void onCardInfo(CardInfoEntity cardInfoEntity) {
            mProgressDialog.dismiss();
            byte rfCardType = cardInfoEntity.getRfCardType();
            readRfCard(rfCardType);
        }


        @Override
        public void onError(int i) {
            mProgressDialog.dismiss();
            showReadRfCardErrorDialog(i);
        }


        @Override
        public void onNoCard(CardReaderTypeEnum cardReaderTypeEnum, boolean b) {

        }
    };
```

```java
    public static final byte[] APDU_SEND_RF = {0x00, (byte) 0xA4, 0x04, 0x00, 0x0E, 0x32, 0x
50, 0x41, 0x59, 0x2E, 0x53, 0x59, 0x53, 0x2E, 0x44, 0x44, 0x46, 0x30, 0x31, 0x00};
    public static final byte[] APDU_SEND_FELICA = {0x10, 0x06, 0x01, 0x2E, 0x45, 0x76, (byte
) 0xBA, (byte) 0xC5, 0x45, 0x2B, 0x01, 0x09, 0x00, 0x01, (byte) 0x80, 0x00};
    private void readRfCard(final byte rfCardType) {
        RfCard rfCard = mCardReadManager.getRFCard();
        int result = rfCard.rfReset();
        if(result == SdkResult.SDK_OK) {
            byte[] apduSend;
            if (rfCardType == SdkData.RF_TYPE_FELICA) { // felica card
                apduSend = APDU_SEND_FELICA;
            } else {
                apduSend = APDU_SEND_RF;
            }
            int[] recvLen = new int[1];
            byte[] recvData = new byte[300];
            result = rfCard.rfExchangeAPDU(apduSend, recvData, recvLen);
            if(result == SdkResult.SDK_OK) {
                final String apduRecv = StringUtils.convertBytesToHex(recvData).substring(0,
 recvLen[0] * 2);
                CardFragment.this.getActivity().runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                        DialogUtils.show(getActivity(), "Read contactless card result",
                                "Card type: " + rfCardTypeToString(rfCardType) + "\n" +
                                "Result: " + apduRecv);
                    }
                });
            } else {
                showReadRfCardErrorDialog(result);
            }
        } else {
            showReadRfCardErrorDialog(result);
        }
        rfCard.rfCardPowerDown();
    }

    private void showReadRfCardErrorDialog(final int errorCode) {
        CardFragment.this.getActivity().runOnUiThread(new Runnable() {
            @Override
            public void run() {
                DialogUtils.show(getActivity(), "Read contactless card failed", "Error code
= " + errorCode);
```

```
            }
        });
    }
```

6. Other cards

For other special cards such as M1 card, mifare plus card, SLE4428 card, SLE4442 card and other cards, please refer to the 'demo' in the attachment. If there are unsupported cards, please contact the business for customization.

7. NFC

The NFC function adopts Android native API. For the specific usage method, please refer to the official API document.

EMV

This chapter only shows the EMV process and key codes. For the complete process, please refer to 'EmvActivity.java' in the attached 'demo'.

Please refer to 'EMV_API.doc' for specific EMV interfaces and parameter meanings.

1. EMV transaction flow chart
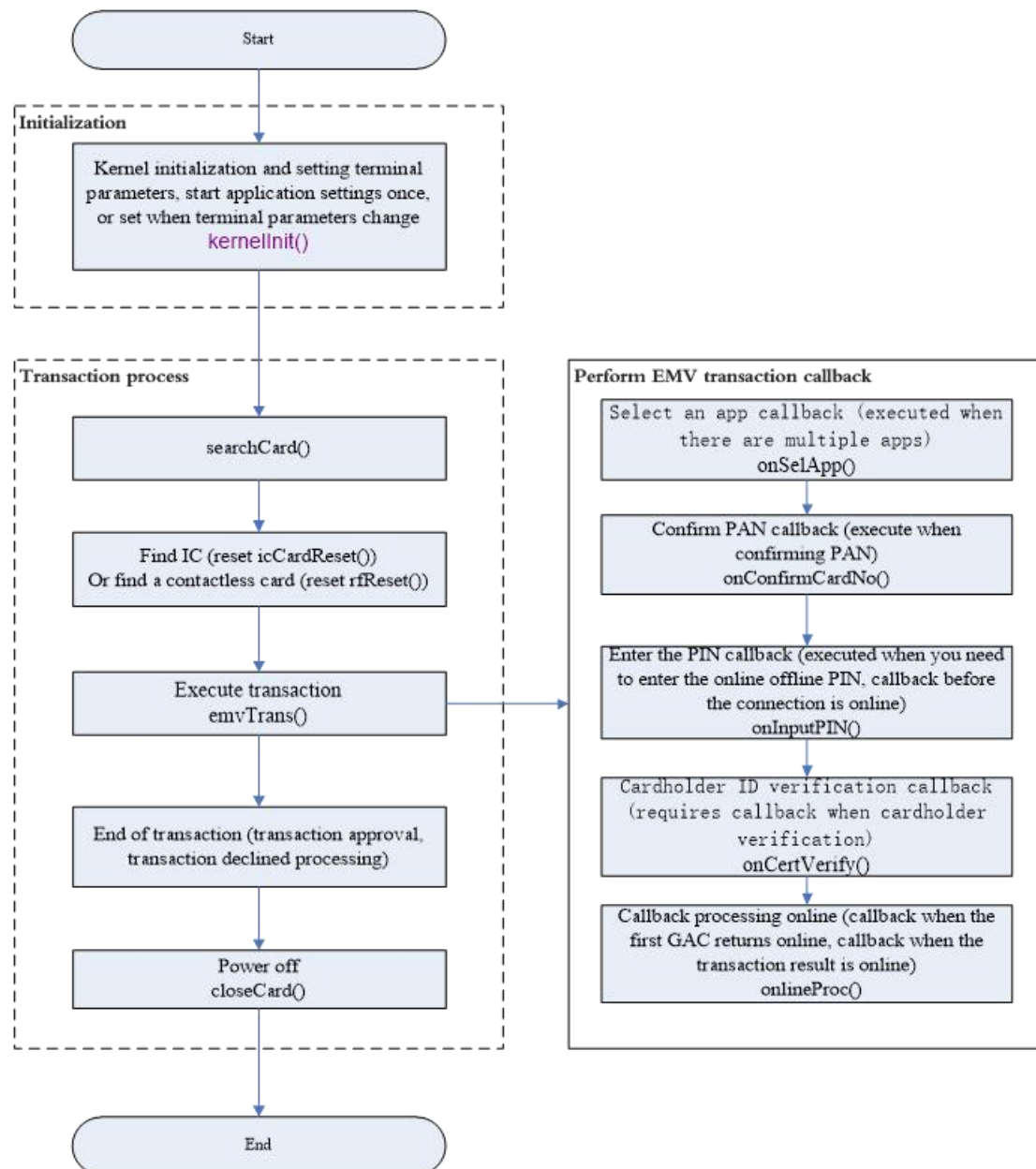
2. Initialization

```
emvHandler = EmvHandler.getInstance();

mPinPadManager = mDriverManager.getPadManager();

byte[] pucIsEcTrans = new byte[1];

byte[] pucBalance = new byte[6];

byte[] pucTransResult = new byte[1];
```

3. Callback

```
OnEmvListener onEmvListener = new OnEmvListener() {
```

```java
    @Override
    public int onSelApp(String[] appLabelList) {
        Log.d("Debug", "onSelApp");
        return 0;
    }


    @Override
    public int onConfirmCardNo(String cardNo) {
        Log.d("Debug", "onConfirmCardNo");
        String[] track2 = new String[1];
        final String[] pan = new String[1];
        emvHandler.getTrack2AndPAN(track2, pan);
        int index = 0;
        if (track2[0].contains("D")) {
            index = track2[0].indexOf("D") + 1;
        } else if (track2[0].contains("=")) {
            index = track2[0].indexOf("=") + 1;
        }
        final String exp = track2[0].substring(index, index + 4);
        showLog("cardNum:" + pan[0]);
        showLog("exp:" + exp);
        return 0;
    }


    @Override
    public int onInputPIN(byte pinType) {
        // 1. open the secret pin pad to get pin block
        // 2. send the pinBlock to emv kernel
        if (emvTransParam.getTransKernalType() == EmvData.KERNAL_CONTACTLESS_ENTRY_POINT) {
            String[] track2 = new String[1];
            final String[] pan = new String[1];
            emvHandler.getTrack2AndPAN(track2, pan);
            int index = 0;
            if (track2[0].contains("D")) {
                index = track2[0].indexOf("D") + 1;
            } else if (track2[0].contains("=")) {
                index = track2[0].indexOf("=") + 1;
            }
            final String exp = track2[0].substring(index, index + 4);
            showLog("card:" + pan[0]);
            showLog("exp:" + exp);
        }
        Log.d("Debug", "onInputPIN");
        int iRet = 0;
```

```java
        iRet = inputPIN(pinType);
        Log.d("Debug", "iRet=" + iRet);
        if (iRet == EmvResult.EMV_OK) {
            emvHandler.setPinBlock(mPinBlock);
        }
        return iRet;
    }

    @Override
    public int onCertVerify(int certType, String certNo) {
        Log.d("Debug", "onCertVerify");
        return 0;
    }

    @Override
    public byte[] onExchangeApdu(byte[] send) {
        Log.d("Debug", "onExchangeApdu");
        if (realCardType == CardReaderTypeEnum.IC_CARD) {
            return mICCard.icExchangeAPDU(CardSlotNoEnum.SDK_ICC_USERCARD, send);
        } else if (realCardType == CardReaderTypeEnum.RF_CARD) {
            return mRFCard.rfExchangeAPDU(send);
        }
        return null;
    }

    @Override
    public int onlineProc() {
        // 1. assemble the authorisation request data and send to bank by using get 'emvHandler.getTlvData()'
        // 2. separateOnlineResp to emv kernel
        // 3. return the callback ret
        Log.d("Debug", "onOnlineProc");
        byte[] authRespCode = new byte[3];
        byte[] issuerResp = new byte[512];
        int[] issuerRespLen = new int[1];
        int iSendRet = emvHandler.separateOnlineResp(authRespCode, issuerResp, issuerRespLen[0]);
        Log.d("Debug", "separateOnlineResp iSendRet=" + iSendRet);
        return 0;
    }
};
```

4. Set parameters

```
final EmvTransParam emvTransParam = new EmvTransParam();
if (cardType == CardReaderTypeEnum.IC_CARD) {
    emvTransParam.setTransKernalType(EmvData.KERNAL_EMV_PBOC);
} else if (cardType == CardReaderTypeEnum.RF_CARD) {
    emvTransParam.setTransKernalType(EmvData.KERNAL_CONTACTLESS_ENTRY_POINT);
}
emvHandler.transParamInit(emvTransParam);
final EmvTermParam emvTermParam = new EmvTermParam();
emvHandler.kernelInit(emvTermParam);
```

5. Adding AID and CAPK

Note that adding AID and CAPK needs to be executed after 'emvHandler.kernelInit(emvTermParam)', and AID and CAPK will be persistently stored in the file system, so it is necessary to avoid repeated additions. The previously added ones can be cleared through 'emvHandler.delAllApp() ' or 'emvHandler.delAllCapk()' .

```
private void loadVisaAIDs(EmvHandler emvHandle) {
    // Visa Credit/Debit EmvApp
    ea = new EmvApp();
    ea.setAid("A0000000031010");
    ea.setSelFlag((byte) 0);
    ea.setTargetPer((byte) 0x00);
    ea.setMaxTargetPer((byte) 0);
    ea.setFloorLimit(1000);
    ea.setOnLinePINFlag((byte) 1);
    ea.setThreshold(0);
    ea.setTacDefault("0000000000");
    ea.setTacDenial("0000000000");
    ea.setTacOnline("0000000000");
    ea.settDOL("0F9F02065F2A029A039C0195059F3704");
    ea.setdDOL("039F3704");
    ea.setVersion("008C");
    ea.setClTransLimit("000000015000");
    ea.setClOfflineLimit("000000008000");
    ea.setClCVMLimit("000000005000");
    ea.setEcTTLVal("000000100000");
    emvHandle.addApp(ea);
```

```
    }

    private void loadMasterCardCapks(EmvHandler emvHandle) {
        EmvCapk capk = new EmvCapk();
        capk.setKeyID((byte) 0x05);
        capk.setRID("A000000004");
        capk.setModul("B8048ABC30C90D976336543E3FD7091C8FE4800"
                + "DF820ED55E7E94813ED00555B573FECA3D84AF6"
                + "131A651D66CFF4284FB13B635EDD0EE40176D8B"
                + "F04B7FD1C7BACF9AC7327DFAA8AA72D10DB3B"
                + "8E70B2DDD811CB4196525EA386ACC33C0D9D45"
                + "75916469C4E4F53E8E1C912CC618CB22DDE7C3"
                + "568E90022E6BBA770202E4522A2DD623D180E21"
                + "5BD1D1507FE3DC90CA310D27B3EFCCD8F83DE"
                + "3052CAD1E48938C68D095AAC91B5F37E28BB49EC7ED597");
        capk.setCheckSum("EBFA0D5D06D8CE702DA3EAE890701D45E274C845");
        capk.setExpDate("20211231");
        // YYYYMMDD
        emvHandle.addCapk(capk);
    }
```

6. Running and result

```
    int ret = emvHandler.emvTrans(emvTransParam, onEmvListener, pucIsEcTrans, pucBalance, pu
cTransResult);
    showLog("Emv trans end, ret = " + ret);
    String str = "Decline";
    if (pucTransResult[0] == EmvData.APPROVE_M) {
        str = "Approve";
    } else if (pucTransResult[0] == EmvData.ONLINE_M) {
        str = "Online";
    } else if (pucTransResult[0] == EmvData.DECLINE_M) {
        str = "Decline";
    }
    showLog("Emv trans result = " + pucTransResult[0] + ", " + str);
    if (ret == 0) {
        getEmvData();
    }
    mCardReadManager.closeCard();

int[] tags = {
        0x9F26,
        0x9F27,
```

```
                0x9F10,
                0x9F37,
                0x9F36,
                0x95,
                0x9A,
                0x9C,
                0x9F02,
                0x5F2A,
                0x82,
                0x9F1A,
                0x9F03,
                0x9F33,
                0x9F34,
                0x9F35,
                0x9F1E,
                0x84,
                0x9F09,
                0x9F41,
                0x9F63,
                0x5F24
    };

private void getEmvData() {
    byte[] field55 = emvHandler.packageTlvList(tags);
    showLog("Filed55: " + StringUtils.convertBytesToHex(field55));
}
```

PinPad

This chapter only shows the key codes of the PinPad. For the complete process, please refer to 'PinpadFragment.java' in the attached 'demo'.

It needs to be initialized before use and obtain the 'PinPadManager 'instance.

```
PinPadManager mPadManager = mDriverManager.getPadManager();
```

1. Set main key

```java
    private void setMainKey() {
        String main_key = "31313131313131313232323232323232";
        byte[] main_key_byte = StringUtils.convertHexToBytes(main_key);
        // index: means Key index, 0x00~0x0F
        // key: The key length is a multiple of 8.
        int ret = mPadManager.pinPadUpMastKey(0,  //index
                main_key_byte ,  //key
                (byte) main_key_byte.length);
        Log.d(TAG, "setMainKey: " + ret);
    }
```

2. Set work key

```java
    private void setWorkKey() {
        String pin_key = "BF1CA957FE63B286E2134E08A8F3DDA903E0686F";
        String mac_key = "8670685795c8d2ea0000000000000000d2db51f1";
        String tdk_key = "00A0ABA733F2CBB1E61535EDCFDC34A93AA3EA2D";

        byte[] pin_key_byte = StringUtils.convertHexToBytes(pin_key);
        byte[] mac_key_byte = StringUtils.convertHexToBytes(mac_key);
        byte[] tdk_key_byte = StringUtils.convertHexToBytes(tdk_key);

        int ret
= pinPadManager.pinPadUpWorkKey(index_all, pin_key_byte, (byte) pin_key_byte.length,
                mac_key_byte, (byte) mac_key_byte.length, tdk_key_byte, (byte) tdk_key_byte.
length);
        Log.d(TAG, "setWorkKey: " + ret);

    }
```

3. Pinblock (secure random PinPad)

To use pinblock, you need to declare the specified 'Activity' in the

'AndroidManifest', and define its style.

```xml
<!-- add in AndroidManifest.xml-->
<activity android:name="com.zcs.sdk.pin.pinpad.PinPadPasswordActivity"
    android:theme="@style/Theme.WindowActivity">
</activity>

<!-- add in styles.xml-->
```

```xml
<style name="Theme.WindowActivity" parent="android:style/Theme.Dialog">
    <item name="android:windowNoTitle">true</item>
    <item name="android:windowIsTranslucent">true</item>
    <item name="android:windowBackground">@android:color/transparent</item>
    <item name="android:windowContentOverlay">@null</item>
    <item name="android:windowIsFloating">true</item>
    <item name="android:backgroundDimEnabled">true</item>
    <item name="android:windowAnimationStyle">@android:style/Animation.Dialog</item>
</style>
```

Used in code.

```java
pinPadManager.inputOnlinePin(getActivity(), (byte) 6, (byte) 12, 60, true, "5187108106590784
", (byte) 0, PinAlgorithmMode.ANSI_X_9_8, new PinPadManager.OnPinPadInputListener() {
        @Override
        public void onError(final int code) {
        }
        @Override
        public void onSuccess(final byte[] bytes) {
            Log.d(TAG, "PinBlock: " + StringUtils.convertBytesToHex(bytes));
        }
    });
```

4. MAC

```java
private void mac() {
    String mac_data = "0200302004C030C09811000000000000000000010000080210001232518710810659076
99B0E751ADD38E068010499518710810659078 4D1561561999999993001999000000003434134130310DD068423601
059800005219298D060D745153979CC00313233343536373831323334353637383930313233343535313536117A7E3
A0DFD4179261000000000000000001422000335000601";
    byte[] mac = new byte[8];
    int ret = mPadManager.pinPadMac(0, PinMacTypeEnum.ECB, StringUtils.convertHexToBytes(mac
_data), mac_data.length() / 2, mac);
    Log.d(TAG, "mac: " + ret + " " + StringUtils.convertBytesToHex(mac));
}
```

5. Encrypt track data

```java
private void encryptTrack() {
    String track = "6258091644092434=20102010000089500000";
    // track is ascii string, one letter is one byte
    // hex string: two letter is one byte
    byte[] encryptedTrack = new byte[track.length()];
```

```
    int ret = mPadManager.pinPadEncryptTrackData(0, MagEncryptTypeEnum.UNION_ENCRYPT, track.
getBytes(), (byte) (track.length()), encryptedTrack);
    Log.d(TAG, "encryptTrack: " + ret + " " + new String(encryptedTrack));
}
```

6. Encrypt data

```
private void encryptData() {
    String dataForDes = "1111111111111111111111111111111";
    byte[] res = new byte[dataForDes.length() / 2];
    int ret = mPadManager.pinPadEncryptData(0, PinWorkKeyTypeEnum.MAC_KEY, StringUtils.conve
rtHexToBytes(dataForDes), dataForDes.length() / 2, res);
    Log.d(TAG, "encryptData: " + ret + " " + StringUtils.convertBytesToHex(res));
}
```

7. About dukpt key

7.1 Set dukpt key

```
private void setDukptKey() {
    String key = "6AC292FAA1315B4D858AB3A3D7D5933A";
    String ksn = "FFFF9876543210E00000";
    int upDukpt = mPadManager.pinPadUpDukpt(0, StringUtils.convertHexToBytes(key), (byte) (k
ey.length() / 2), StringUtils.convertHexToBytes(ksn));
}
```

7.2 Get pinblock

```
private void getPinBlockByDukpt() {
    final byte[] ksn = new byte[10];
    mPadManager.inputOnlinePinByDukpt(getActivity(), (byte) 6, (byte) 12, 60, true, "5187108
106590784", (byte) 0, PinAlgorithmMode.ANSI_X_9_8, new PinPadManager.OnPinPadInputListener()
 {
        @Override
        public void onError(final int code) {
        }

        @Override
        public void onSuccess(final byte[] bytes) {
            Log.d(TAG, "PinBlock: " + StringUtils.convertBytesToHex(bytes));
            Log.d(TAG, "ksn: " + StringUtils.convertBytesToHex(ksn));
```

```
        }
    }, ksn);
}
```

7.3 MAC

```java
private void getMacByDukpt() {
    String input = "0200302004C030C09811000000000000000000010000080210001232518710810659076998
0E751ADD38E06801049951871081065907840D156156199999993001999000000343434130310DD0684236010590
800005219298D060D745153979CC003132333435363738313233343536373839303132333435313536117A7E3A0D
FD417926100000000000000000001422000335000601";
    byte[] outData = new byte[8];
    byte[] ksn = new byte[10];
    int ret = mPadManager.pinPadMacByDukpt(0, PinMacTypeEnum.ECB, StringUtils.convertHexToBy
tes(input), input.length() / 2, outData, ksn);
    Log.d(TAG, "pinPadMacByDukpt:" + ret);
    if (ret == SdkResult.SDK_OK) {
        Log.d(TAG, "outData:" + StringUtils.convertBytesToHex(outData));
        Log.d(TAG, "ksn:" + StringUtils.convertBytesToHex(ksn));
    }
}
```

7.4 Encrypt

```java
private void encryptByDukpt() {
    String input = "0200302004C030C09811000000000000000000010000080210001232518710810659076998
0E751ADD38E06801049951871081065907840D156156199999993001999000000343434130310DD0684236010590
800005219298D060D745153979CC003132333435363738313233343536373839303132333435313536117A7E3A0D
FD417926100000000000000000001422000314220003";
    byte[] outData = new byte[input.length() / 2];
    byte[] ksn = new byte[10];
    int ret = mPadManager.pinPadEncryptDataByDukpt(0, PinWorkKeyTypeEnum.PIN_KEY, StringUtil
s.convertHexToBytes(input), input.length() / 2, outData, ksn);
    Log.d(TAG, "sdkPadEncryptDataByDukpt:" + ret);
    if (ret == SdkResult.SDK_OK) {
        Log.d(TAG, "outData:" + StringUtils.convertBytesToHex(outData));
        Log.d(TAG, "ksn:" + StringUtils.convertBytesToHex(ksn));
    }
}
```

Others

This chapter includes the interfaces of hardware-related functions, all of which require hardware support for normal use. If the hardware does not support it, an error code will be returned.

1. LED

```
int ret = mLed.setLed(LedLightModeEnum.RED, true);
```

2. Beeper

```
int ret = mBeeper.beep(4000, 600);
```

3. Cutter

```java
private void cutPaper() {
    int printStatus = mPrinter.getPrinterStatus();
    if(printStatus == SdkResult.SDK_OK) {
        mPrinter.openPrnCutter((byte) 1);
    }
}
```

4. LED customer display

Only supports bitmap of size 128px*64px.

```
mSys.showBitmapOnLcd(bitmap, true);
```

5. Cash drawer

```
mPrinter.openBox();
```

6. Scanner

The scanner needs to be powered on before it can be used.

(There may be slight differences in the code for different models of

Scanner. If there are any issues during development, please contact the

business department.)

6.1 Power on

```
mhqscanner = mDriverManager.getHQrsannerDriver();
mhqscanner.QRScanerPowerCtrl((byte) 1);
```

6.2 Open

```
mhqscanner.QRScanerCtrl((byte)1);
try {
    Thread.sleep(10);
} catch (InterruptedException e) {
    e.printStackTrace();
}
mhqscanner.QRScanerCtrl((byte)0);
```

Other considerations

Do not obfuscate the code in the sdk when compiling, add the

following configuration to the obfuscation configuration.

```
-keep class com.zcs.base.SmartPosJni{*;}
-keep class com.zcs.sdk.DriverManager{*;}
-keep class com.zcs.sdk.emv.**{*;}
```