POS simple development tutorial

Foreword

Currently, the API provided by the POS is the C interface. Please see Interface Document file: \ DeviceDrivers \ CLOUDPOS_SDK_Development _Guide_EN.pdf, which details how to use the interface. Third-party applications can directly use the underlying "C interface" for development. In addition, in order to facilitate customers who don't have development experience of language C, we provide the JNI interface, to which please refer: \ DeviceDrivers \ APIDemo, the JNI source packaged the calling of C interface, and third-party applications can use the JNI interface directly, just copy the corresponding "SO File" and "JNI Interface" and paste to the place below their own projects. In addition, the JNI package code is open sourced, so users who have basic knowledges of C can modify the "JNI" code and re-package. This article is mainly to help users use the packaged JNI interface, create and run their own projects.

The SimplePaymentDemo app demoed in the document is in
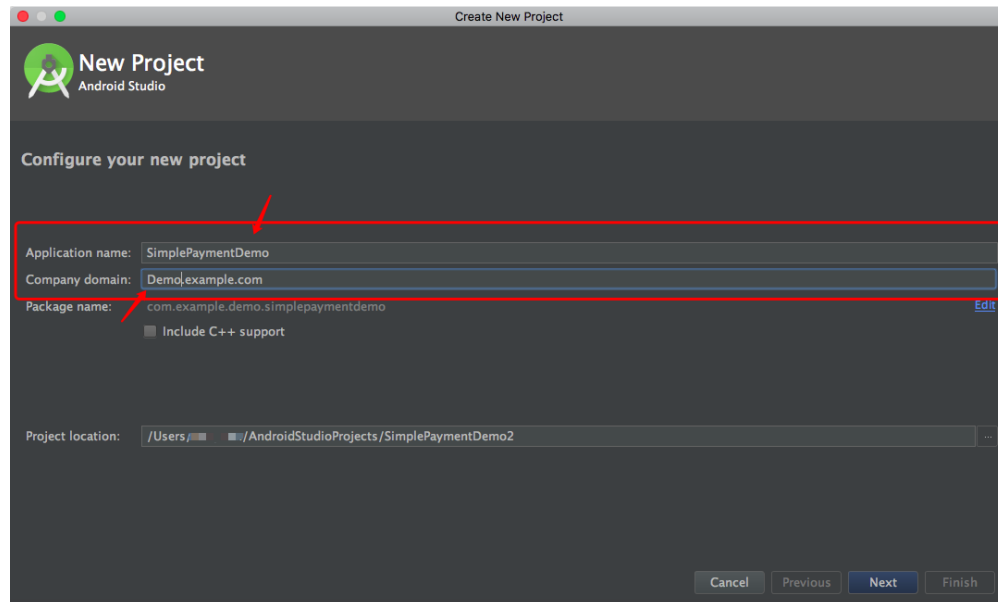
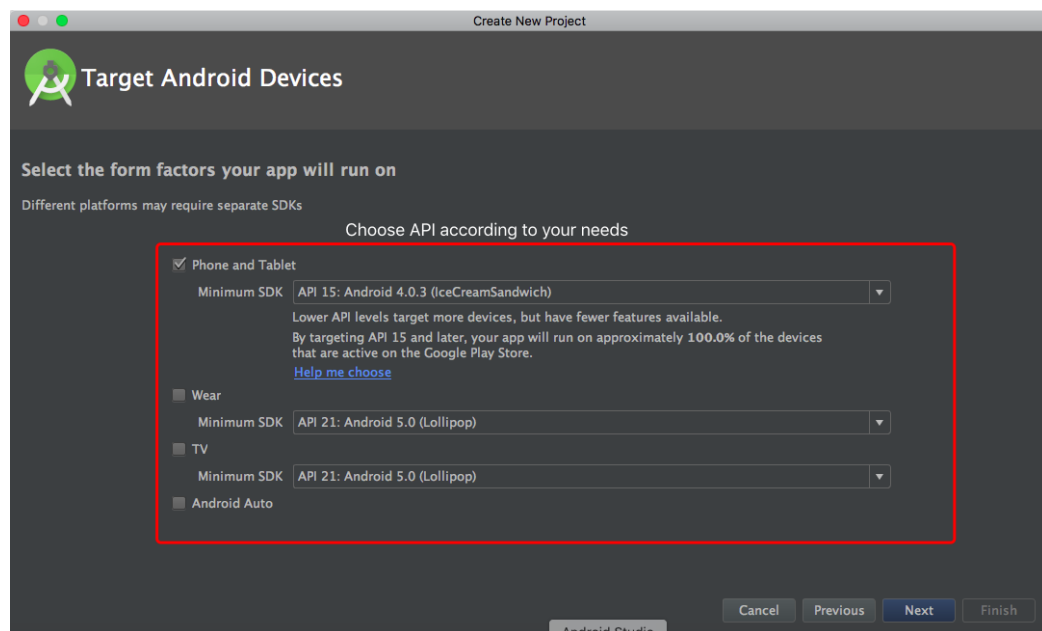\DeviceDrivers\SimpleExamples\SimplePaymentDemo.

Contents

1. Project creation

1.1 Engineering creation

Create a new project, name "SimplePaymentDemo",process as shown in the following picture:
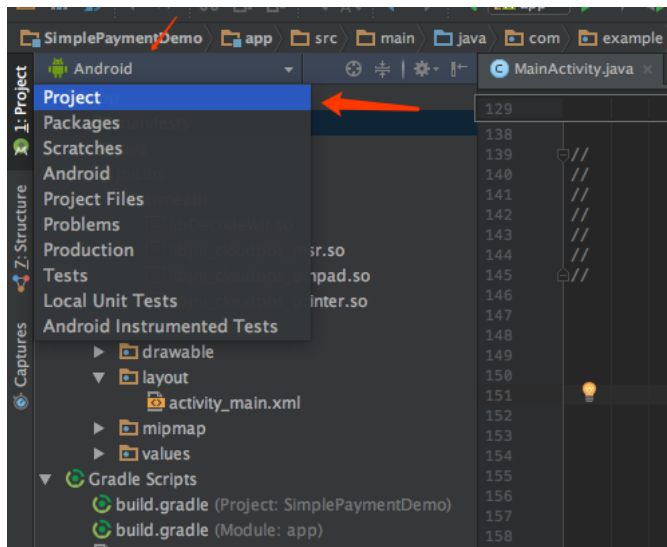

Then click "Next".

Then click "Next".

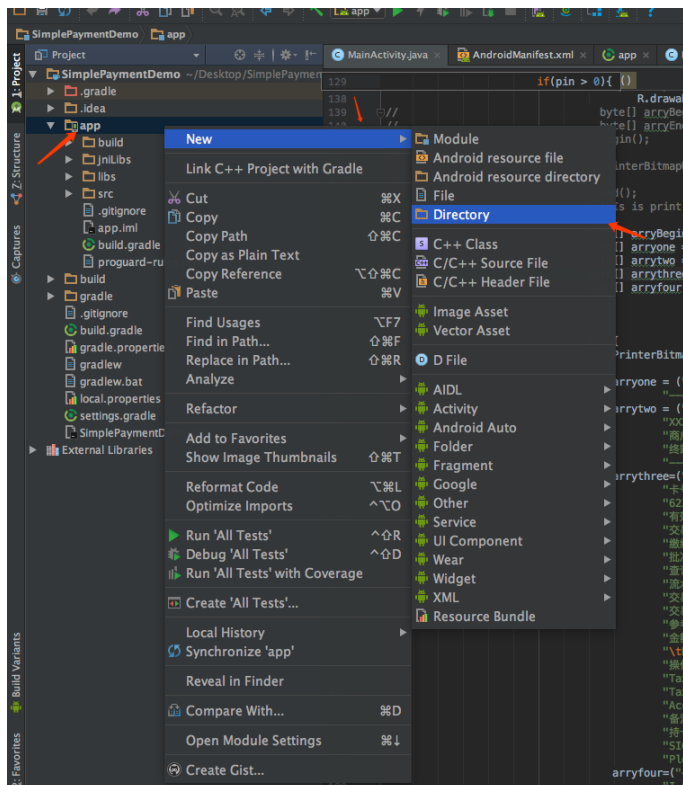Then leave all the defaults, and click "Finish" at last.

1.2 Import related files

Please copy the relevant ".so" file and "jniinterface", this project needs to copy the magnetic stripe card, password keyboard, printer part.

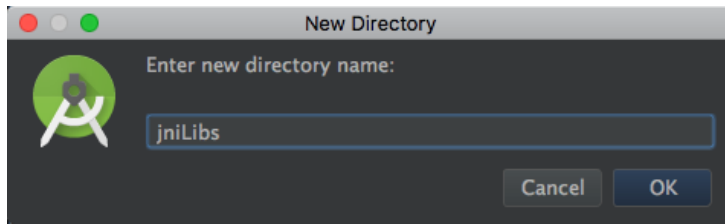As shown, click the "Android" drop-down menu and select "project".

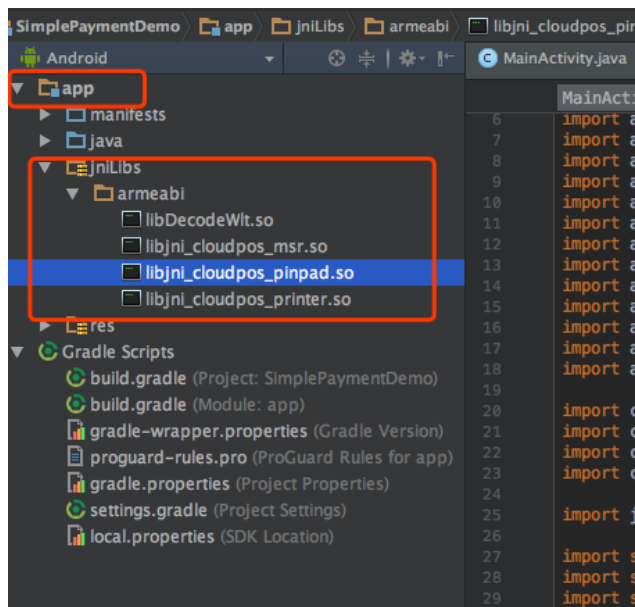Right click at app, and click New - Directory.

The folder name is called "jniLibs" (must be exactly the same).
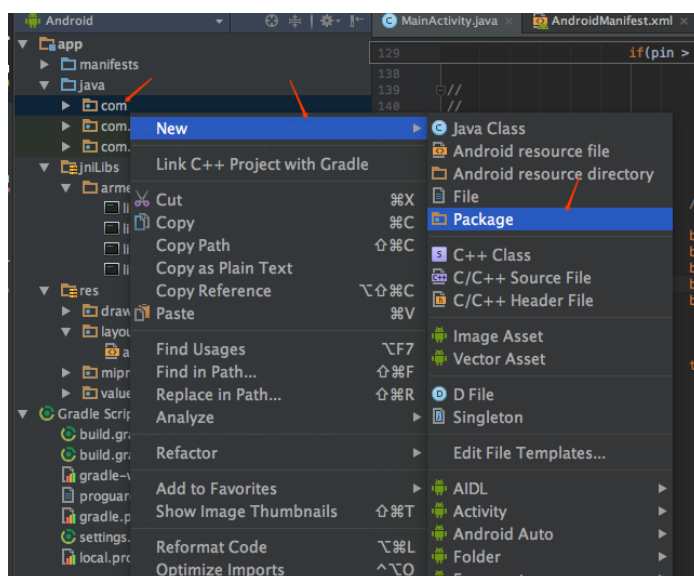
And then put the .so file we need inside (Note: folder armeabi should also be established, with specific procedures taken as a step forward).

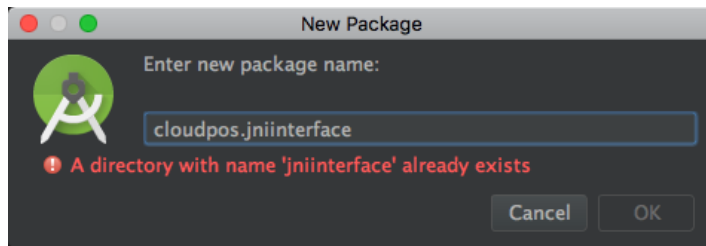
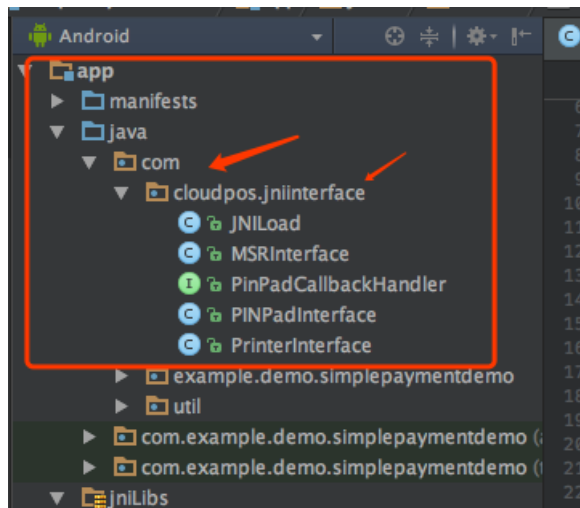In the src/main/java/com/ directory,create a new package called"cloudpos.jniinterface".

Because I have established it here, there is a red letter that reminds me that I already exist.

And copy the required "interface" file into it.

Then click on the menu bar "Build" -- "Clean Project".

Normally, in the Gradle file, the "jniLibs" path code will automatically generate after click the "clean project".

if you do not generate it automatically, add the following code manually.

```
sourceSets {

    main {

        jniLibs.srcDirs = ['jniLibs']

    }

}
```

1.3 Permission adding

Add the following permissions in Android manifest.XML.

```
<!—MSR Limit -->

<uses-permission android:name="android.permission.CLOUDPOS_MSR"/>

<!—Pinpad Limit -->

<uses-permission android:name="android.permission.CLOUDPOS_PIN_GET_PIN_BLOCK"/>
```

```
<uses-permission android:name="android.permission.CLOUDPOS_PIN_MAC" />

<uses-permission android:name="android.permission.CLOUDPOS_PIN_ENCRYPT_DATA"/>

<uses-permission android:name="android.permission.CLOUDPOS_PIN_UPDATE_USER_KEY"/>

<uses-permission android:name="android.permission.CLOUDPOS_PIN_UPDATE_MASTER_KEY"/>

<!—Print Limit -->

<uses-permission android:name="android.permission.CLOUDPOS_PRINTER" />
```

2. Writing related business logic

UI 1, prompt credit card interface, interface display, please swipe.

UI 2, swipe card displays card number, validity, input amount, confirmation key.

UI 3, prompted to enter the password, password, enter the password after the keyboard.

UI 4, display transaction completed, print small ticket.

3. Compilation and other matters

3.1 Gradle

The tool we're using is "Android Studio",The gradle content of the project is as follows:

```
apply plugin: 'com.android.application'


android {

    compileSdkVersion 26

    buildToolsVersion "26.0.1"

    defaultConfig {

        applicationId "com.example.demo.simplepaymentdemo"

        minSdkVersion 15

        targetSdkVersion 19

        versionCode 1

        versionName "1.0"

        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
```

```
    }

    buildTypes {

        release {

            minifyEnabled false

            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'

        }

    }


    sourceSets {

        main {

            jniLibs.srcDirs = ['jniLibs']

            aidl.srcDirs = ['src/main/aidl']

        }

    }

}


dependencies {

    compile fileTree(dir: 'libs', include: ['*.jar'])

    androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2', {

        exclude group: 'com.android.support', module: 'support-annotations'

    })

    compile 'com.android.support:appcompat-v7:26.+'

    compile 'com.android.support.constraint:constraint-layout:1.0.2'

    testCompile 'junit:junit:4.12'

}
```
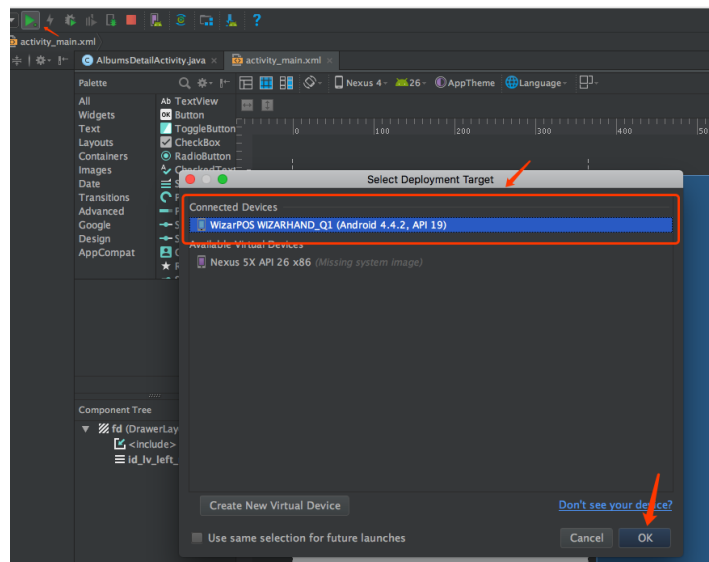
3.2 The signature of the secret key

The operation of "PINPAD" need to inject secret keys (Note: If the master key and user key already exists in the PINPAD, please ignore this step).
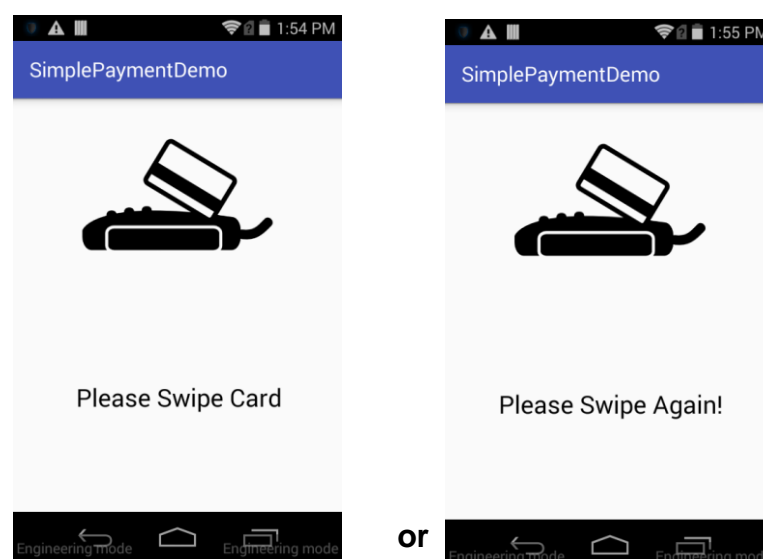
4. Function

Click the green running arrow on the menu bar, and then select the connected device.
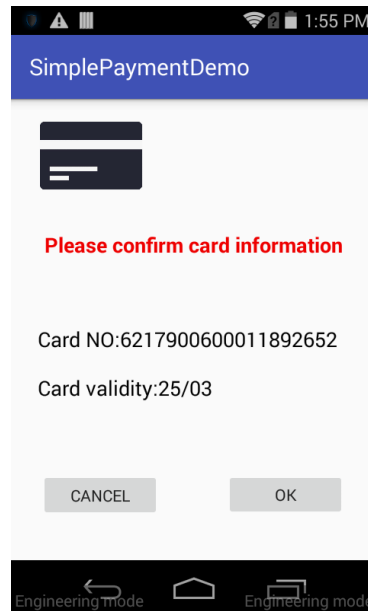

4.1 Operation effect

4.1.1 swipe card UI

Swipe card UI, swipe successfully will jump the next UI, failure will prompt "Please Swipe Again".
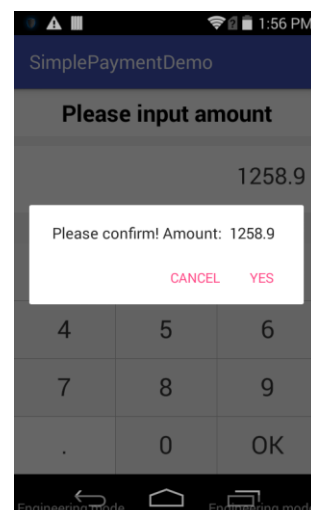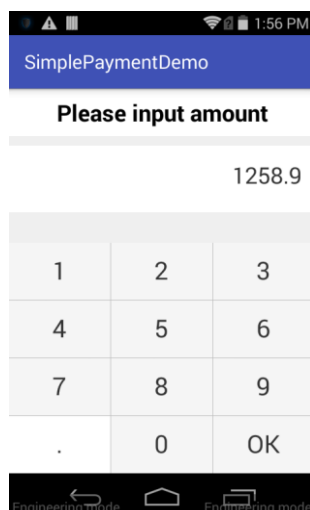
4.1.2 Check the card information and confirm UI

Swipe card will jump to this UI, display card information. Click OK to jump to the next UI.
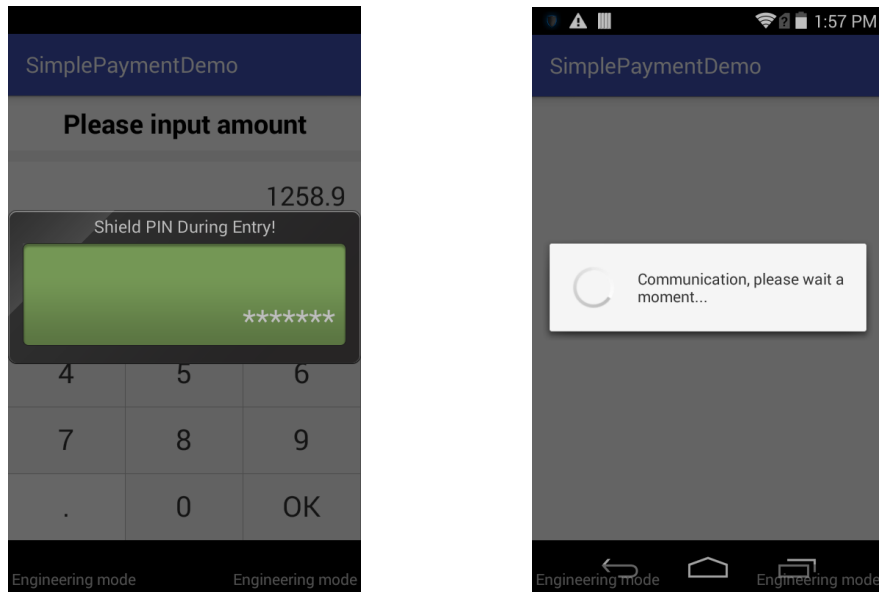

4.1.3 Amount input UI

In the amount input UI, click OK will popup the UI of amount confirmation. Click YES to confirm.

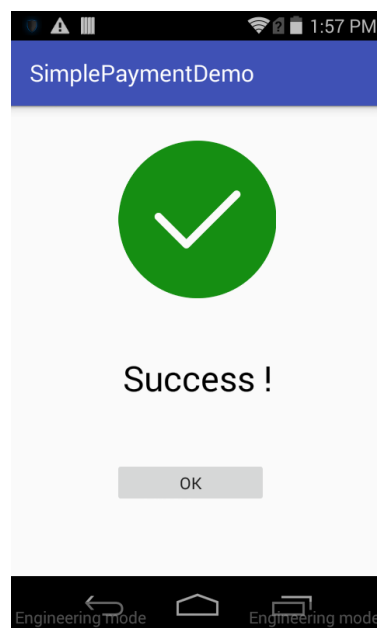### 4.1.4 System page: PINPAD and communication UI

After confirmation of the amount will pop up "PINPAD" input UI, after input pin, will popup communication UI**.**

4.1.5   Payment success UI

After communication, prompt success, print receipt, click OK to finish.

4.1.6 Transaction receipt