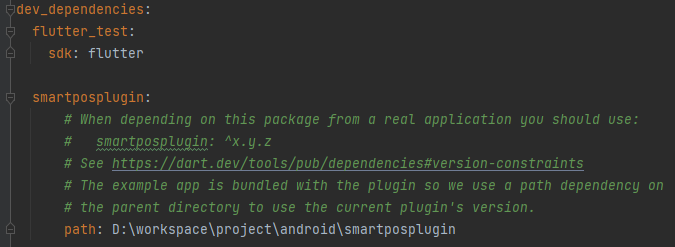
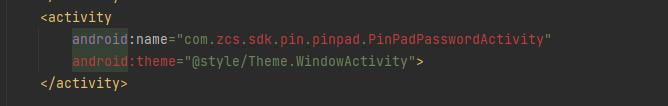
# Smartpos flutter API Instructions

V1.0

Steps to use the plugin,import the plugin into the project and configure the path of the plugin in the pubspec.yaml file, as shown below:



Configure read and write permissions for internal storage in the project and register the password keyboard activity as shown below:



The specific implementation of the interface call, please refer to the file “smartposplugin\example\lib\main.dart”

**Instantiation of The plug-in interface**

Smartposplugin smartpos =Smartposplugin.getInstance();

**API**

|  |  |  |
| --- | --- | --- |
| Methods | Function Description |  |
| Future<String> sysInit() | Communication initialization interface, which must be called before other interfaces can be called,return ‘0’is success otherwise fail |  |
| Future<String> getPid() | Get SN of the device |  |
| Future<String>getFirmwareVer() | Gets the firmware version number of the device |  |
| Future<String>iccRest(int slot) | IC card reset，slot=0 |  |
| Future<String> iccExchangeAPDU(int slot,Uint8List bytes) | IC card apdu exchanges instructions slot=0， bytes =apdu |  |
| Future<String>iccPowerDown(int slot) | IC card halt |  |
| Future<String>iccGetStatus(int slot) | Get the slot status of IC card slot=0 |  |
| Future<String> rfRest() | RF card reset interface |  |
| Future<String> rfSearchCard(int cardType) | Search Rf card ，cardType =0x01 type A(ISO14443A) ，cardType =0x02 type B(ISO14443B)，can be combined,eg: cardType=0x01|0x02 It means that search Type A and Type B card |  |
| Future<String> rfExchangeAPDU(Uint8List sendApdu) | RF IC card apdu exchanges instructions |  |
| Future<String> rfPowerDown() | RF card halt |  |
| Future<String> magClose() | Close Magnetic card |
| Future<String> magIfBrush() | Check whether the magnetic card is brushed |  |
| Future<String> magClearData() | Clear the magnetic card cache data |  |
| Future<String> magReadData() | This interface is called after reading the magnetic card data and detecting the swipe action |  |
| Future<String>prnBitmap(Uint8List pucBitmap) | Print picture ，the parameters of pucBitmap is an array of bmp format |  |
| Future<String>prnStr(Uint8List pucStr) | Print the string interface, the parameters of pucstr is an ASCII string |  |
| Future<String>getPrnStatus() | Get the status of the printer and return "-1403" it means that no paper |  |
| Future<String>prntEpson(Uint8List pucStr) | ESC/POS to print |  |
| Future<String>addCapk(EmvCapk capk) | Add EMV parameters - Add CAPK |  |
| Future<String> addApp(EmvApp aid) | Add EMV parameters - Add AID |  |
| Future<String> delAllCapk() | Delete all CAPK |  |
| Future<String> delAllAid() | Delete all AID |  |
| Future<String> separateOnlineResp(String pucStr) | The parameter is the online response data returned by the bank server |  |
| Future<String> pinPadUpMastKey(int keynum,String ckeyStr) | Download Master key |  |
| Future<String> pinPadUpWorkKey(int keynum,String pin\_keystr, String mac\_keystr,String tdk\_keystr) | Download pin key, mac key and encrypt data key |  |
| Future<String>emvTrans(int cardType) | The execution flow of EMV, cardType= 0 executes contact IC cards and cardType=1 executes contactless IC cards |  |
| Future<String> getEmvData() | This is called after the EmvTans function has been executed to get the Emv 55 domain data |  |

Note ,if you need to use the EMV kernel, you need to implement the following interface to receive kernel callbacks, as described in demo

static const platform = MethodChannel('smartposplugin');

@override

void initState() {

// TODO: implement initState

super.initState();

platform.setMethodCallHandler((call) async {

if (call.method == 'onSelApp') {

platform.invokeMapMethod('messageCall','onSelApp');

String result = call.arguments;

print(result);

}else if(call.method == 'onConfirmCardNo'){

platform.invokeMapMethod('messageCall','onConfirmCardNo');

String result = call.arguments;

print(result);

}else if(call.method == 'onInputPIN'){

platform.invokeMapMethod('messageCall','onInputPIN');

String result = call.arguments;

print(result);

}else if(call.method == 'onCertVerify'){

platform.invokeMapMethod('messageCall','onCertVerify');

String result = call.arguments;

print(result);

}

else if(call.method == 'onlineProc'){

String result = call.arguments;

print(result);

String respcode='Z1';

String issuerResp='8D238A029F02069F03069F1A0295055F2A029A039C019F3704';

String OnlineResp='$respcode;$issuerResp';

platform.invokeMapMethod('messageCall','onlineProc:$OnlineResp');

}

else{

platform.invokeMapMethod('messageCall','');

}

});