I9 SDK INTEGRATION GUIDE

Shenzhen Itron Electronics Co., Ltd.

Version	Date	Author	Description
V1.0	2019/7/24	Robert	Release
V1.02	2019/8/20	Robert	Release
V1.03	2019/9/16	Robert	Release
V1.05	2019/9/29	Beker	Release
V1.06	2019/9/30	Beker	Release
V1.07	2020/3/11	Robert	Add TR-31, Add IOS method
V1.08	2020/3/16	Robert	Add debug keys
V1.09	2020/3/18	Michael	Document layout
V1.10	2020/6/1	Robert	Document layout

Catalogue:

1. Introduction....6
1.1 Summary....6
1.2 Purpose and Scope....7
1.3 Firmware Hardcode Transaction Keys....7
1.4 Key Management and Key Loading....8
1.5 TR-31 Key Loading....10
1.6 EMVCo Terminal Type Approval....12
1.7 Message Format....12
1.8 EMV Standard Tags....12
1.9 Proprietary Tags Description....12
1.10 How to Get I9 Bluetooth Name....12
1.11 How to Connect I9 through Bluetooth....12
2. Transaction....13
2.1 EMV Transaction Process....14
2.2 Magnetic Stripe Transaction....15
3. Development Environment....15
3.1 System And IDE....15
3.2 Android Studio Configure....16
3.3 IOS XCode Configure....16
3.4 Android Permission....17
3.5 IOS Permission....18
3.6 Principle in SDK Methods....19
3.7 Initialized Device....19
3.7.1 Scan Bluetooth Devices....20
3.7.2 Stop Scan Bluetooth Devices....20
3.7.3 Connect Bluetooth Device....21
3.7.4 Disconnect Bluetooth Device....21
3.7.5 Release Bluetooth Resource....22

3.8 Optional Functions..... 22
3.8.1 Terminal Info Function..... 22
3.8.2 Set Terminal Automatic Shutdown Function..... 23
3.8.3 Update Terminal Time Function..... 23
3.8.4 Get Terminal Time..... 24
3.8.5 Operating Stop Function..... 24
3.8.6 Calculate MAC..... 25
3.8.7 PIN Encrypt..... 25
3.8.8 Get Card Number..... 26
3.8.9 Set Buzzer..... 26
3.8.10 Control LED Light..... 26
3.8.11 Get Device Battery Capacity..... 27
3.9 EMV Functions..... 27
3.9.1 Download AID Parameters..... 27
3.9.2 Clear Terminal AID Parameters..... 28
3.9.3 Download CA Public Key..... 29
3.9.4 Clear CA Public Key..... 30
3.9.5 Start EMV Process..... 30
3.9.6 EMV Online Respose..... 31
3.10 APDU Transparent Transfer Functions..... 32
3.10.1 Open APDU Transparent Transfer Function..... 32
3.10.2 Send APDU Packet..... 33
3.10.3 Close APDU Transparent Transfer Function..... 33
3.10 Key Loading Functions..... 34
3.11.1 Master Key / Fixed Key /DUKPT IPEK Injection..... 34
3.11.2 Download Session Key..... 34
4. Listener Functions..... 35
4.1 onGetTerminalInfo..... 35
4.2 onGetCardNo..... 35
4.3 onCaculateMac..... 35
4.4 onPinEncrypted..... 36
4.5 onTimeOut..... 36

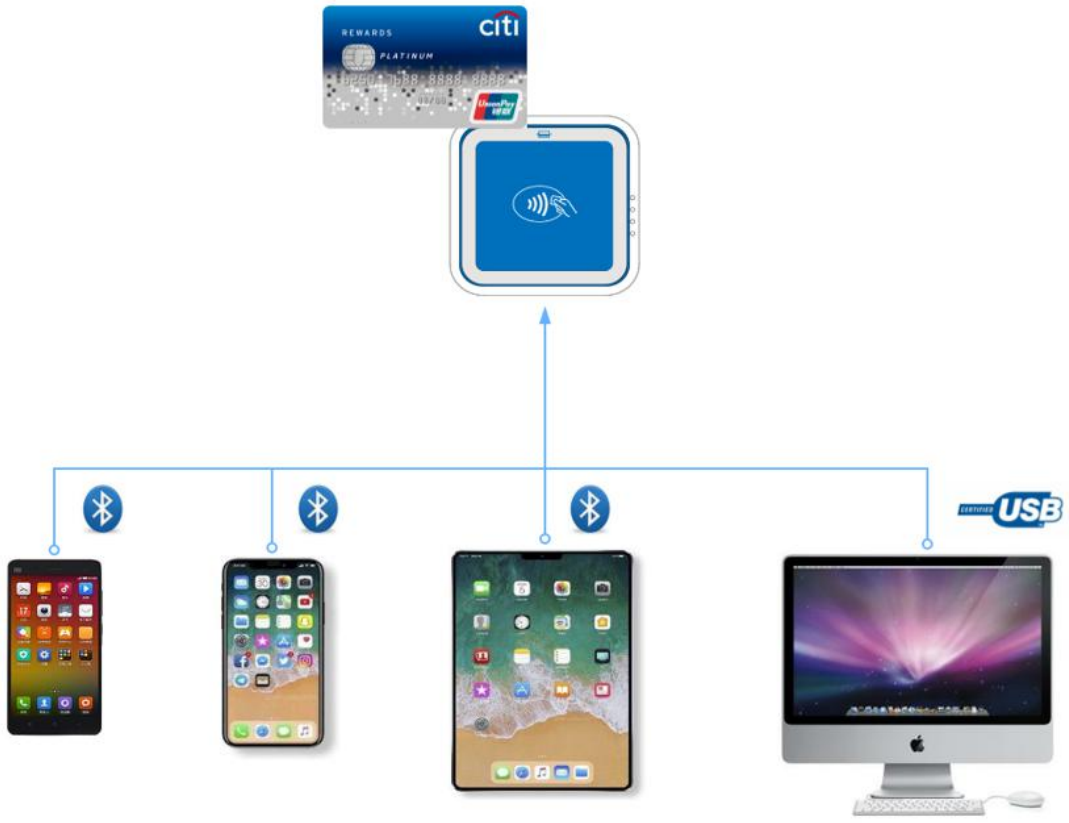
4.6 onError....37
4.7 onWaitingcard....37
4.8 onWaitingPin....38
4.9 onICWaitingOper....38
4.10 onReadCard....39
4.11 onStopTradeSuccess....40
4.12 onExecuteSuccess....41
4.13 onAPDUResponsedata....41
4.14 onBatteryPower....42
4.15 onTerminalDateTime....42
5. Constant Class and Enum....43
5.1 Customer(enum)....43
5.2 ErrorCode(classs)....43
5.3 TradeTag(class)....43
6. Error Code....44

1. Introduction

1.1 Summary

This document is used to guide programmers quickly develop commercial smart phone applications integrated i9 APIs.

I9 connection to a smartphone to form a mobile payment system that provides merchants and consumers a safe and convenient way to process the transaction from mobile environments.



1.2 Purpose and Scope

This document is to describe the APIs of I9. The goal of the I9 Android SDK is to communicate between the smart device and I9.

The readers of the document are those who plans to use I9 android SDK in their application.

1.3 Firmware Hardcode Transaction Keys

Below keys was hardcoded in debug firmware, all default key was index 0 of key type

DUKPT, KSN, Index 0
FFFF9876543210E00000

DUKPT, IPEK, Index 0
6AC292FAA1315B4D858AB3A3D7D5933A

Fixed Key, DES FXK AES, Index 0
11111111111111111111111111111111

Fixed Key, MAC FXK AES, Index 0
22222222222222222222222222222222

MK/SK, Master Key, TDES, Index 0
444444444444444444444444444444444444444444444444

MK/SK, DES Session Key, TDES, Index 0
555555555555555555555555555555555555555555555555

MK/SK, MAC Session Key, TDES, Index 0
666666666666666666666666666666666666666666666666

1.4 Key Management and Key Loading

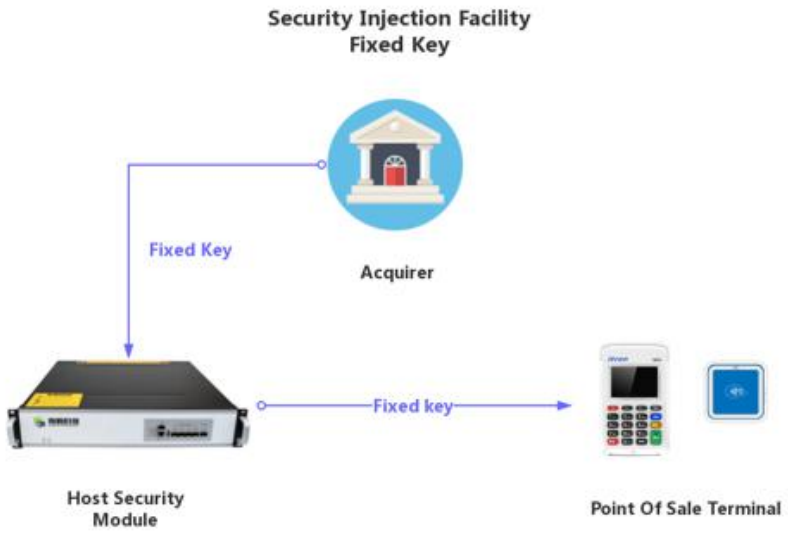
The key-management techniques is implemented in the device and conform to ANSI X9.24. Key-management techniques, which support the ANSI TR-31 key-derivation methodology or an equivalent methodology for maintaining the TDEA key bundle.

Devices support below key management:

- Fixed Transaction Keys
- Master Keys /Session Keys(Transaction Keys)
- Derived Unique Key Per Transaction (DUKPT)

Fixed Key:

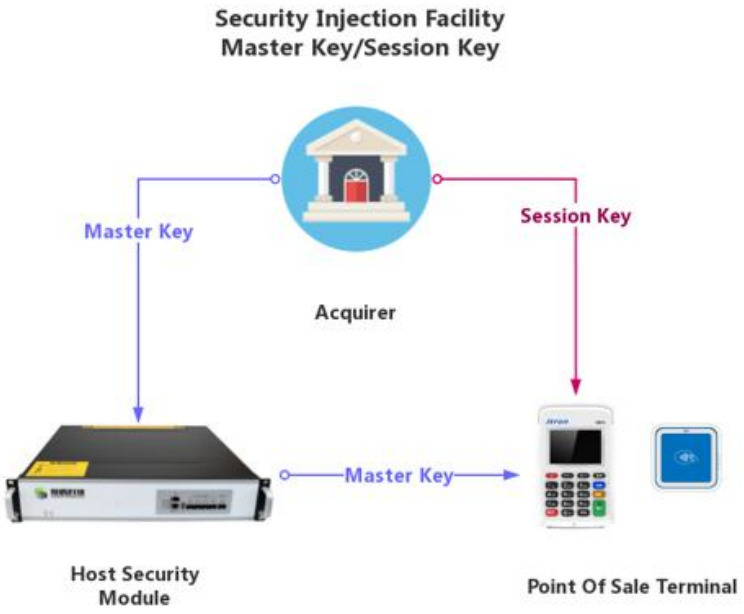
The Fixed Transaction keys should inject to devices before to initiate a transaction.



Master Keys:

The Master keys should inject to devices before deployment machine.

The Session keys (Transaction keys) by Master key inject to devices before to initiate a transaction.



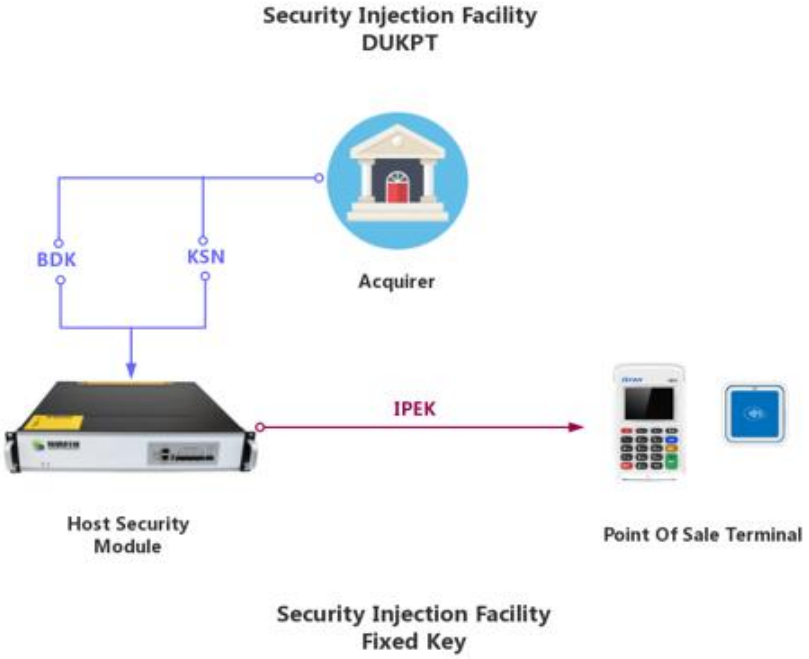
DUKPT:

DUKPT (Derived Unique Key Per Transaction) ensures that a different key is used for every transaction.

A DUKPT Key consists of two parts:

BDK (Base Derivation Key), the working key that is used for encryption

KSN (Key Serial Number), the unique serial number that is injected into each devices.



Host Security Machine deriving the PIN encryption key(IPEK) by BDK and KSN initially loaded in the devices before deployment machine.

For more technical details, please refer to ***<X9.24-1 Retail Financial Services Symmetric Key Management Part 1: Using Symmetric Techniques.>***

1.5 TR-31 Key Loading

Financial terminals often use symmetric key technology to encrypt sensitive data.

The symmetric key of the acquirer must be consistent with the terminal to ensure that the terminal's encrypted data can be correctly decrypted and used by the acquirer.

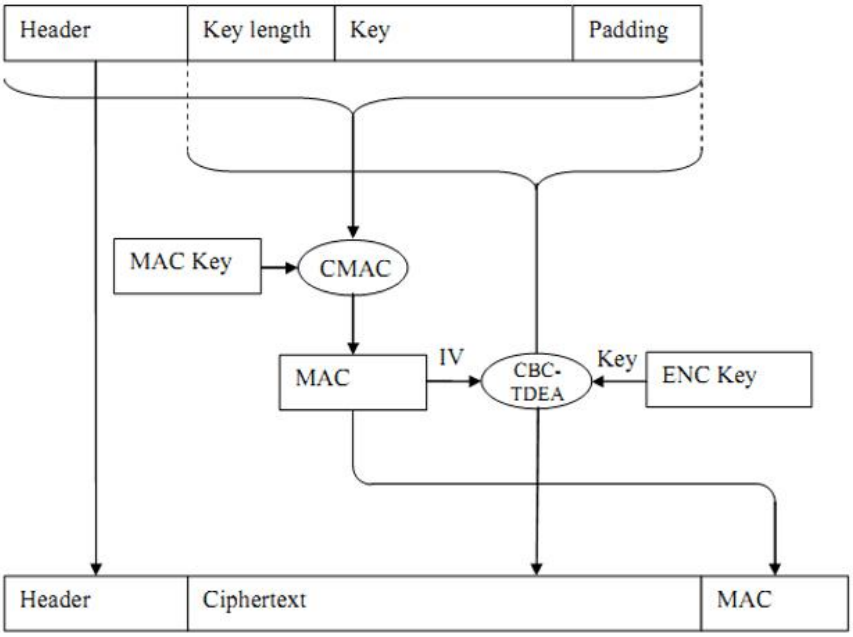
The symmetric key of the acquirer is generally derived from the security device to the financial device terminal. Each key has different uses and algorithms. Then, the security of the symmetric key exchange will be involved.

The TR-31 standard specifies a method in which two devices use a shared symmetric key to exchange other symmetric keys, in accordance with the requirements of ANS X9.24 Retail Financial Services Method Symmetric Key Management Part 1.

The TR-31 mechanism can guarantee:

- The confidentiality of the key
- Key integrity
- Key algorithm
- Purpose of the key
- Key length

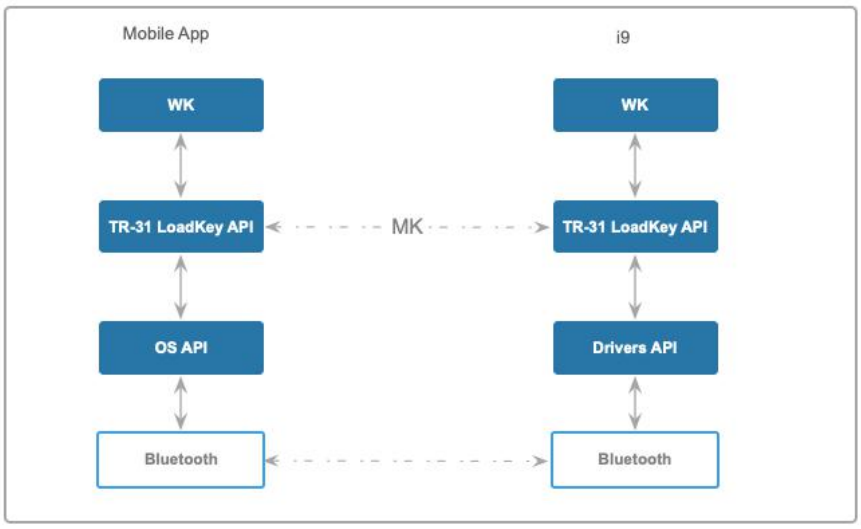
The key download uses the TR-31 scheme, which can ensure the legality of key elements, from this key can ensure the security of the usage.



TR-31 Key Block Binding Method

Itron API provides an interface that conforms to the TR-31 key security Binding. When calling the API, the key attributes and KPBK are passed in, and the data after the Key Binding is output.

Itron TR-31 API supports DUKPT, Fixed Key and MK / SK key system, which can securely pack TDES 128bits or 192bits, AES (128bits) keys.



TR-31 Key Loading Flow

1.6 EMVCo Terminal Type Approval

EMVCo has approved i9 application EMVCo type for Terminal level 2. i9 application is based on the requirements stated in the EMV 4.3 specification.

1.7 Message Format

Messages within data communication protocols between the mobile payment application and i9 EMV kernel are encoded as a BER-TLV (Basic Encoding Rules-Tag-Length-Value) which is defined in EMV 4.3 book3 Annex B.

1.8 EMV Standard Tags

EMV Standard Tags are defined in EMV 4.3 book3 Annex A.

1.9 Proprietary Tags Description

Data format is Tag Length Value format.

1.10 How to Get I9 Bluetooth Name

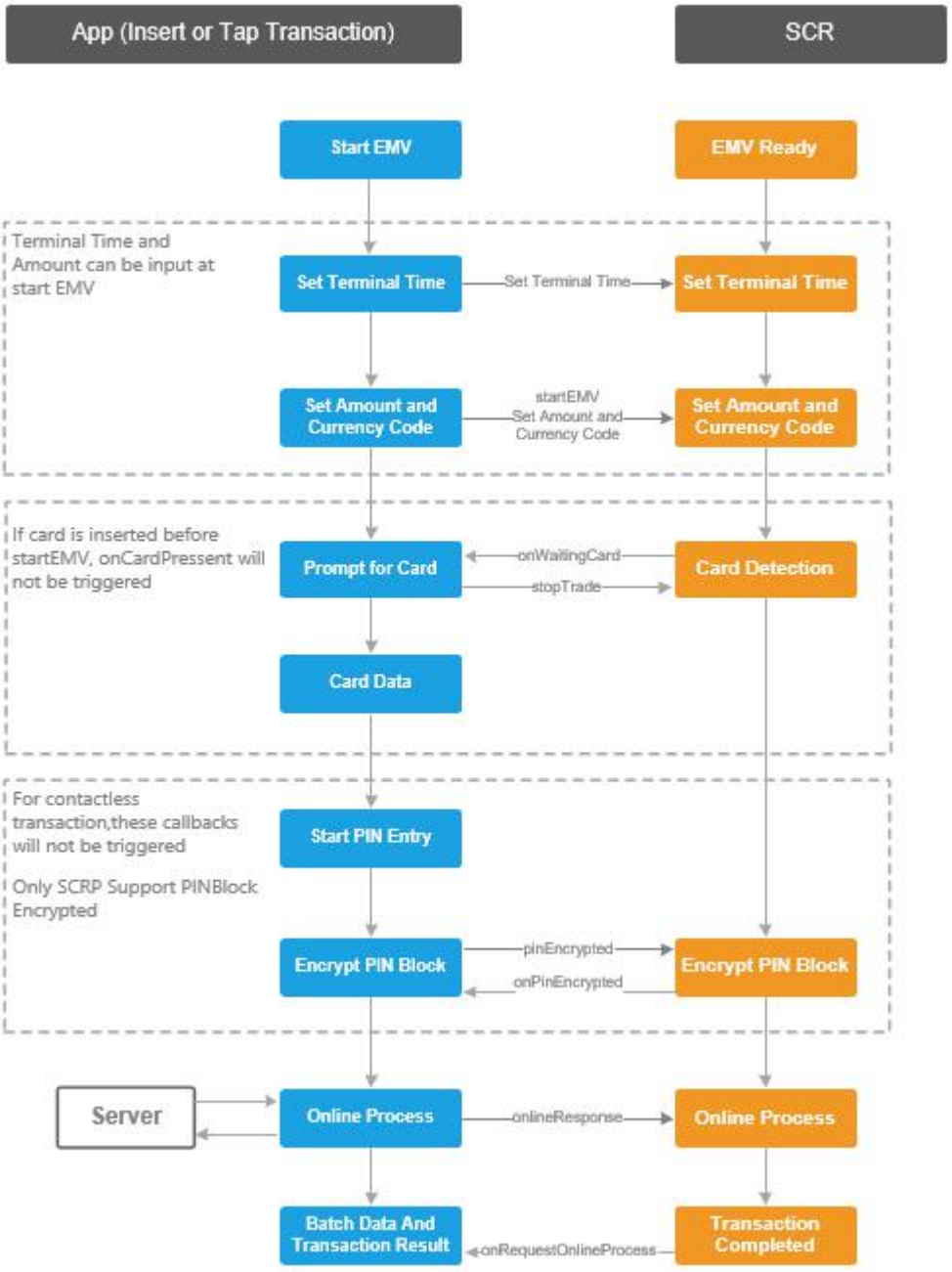
i9's Bluetooth name consists of the last 4 digits of "TSN" and "it"
TSN print on a label that stick to the packing box.
For Example:
TSN = XXXXXXXXXXXXXXXXDB20
Bluetooth name = itDB20

1.11 How to Connect I9 through Bluetooth

Bluetooth only supports BLE mode, don't support SPP.

2. Transaction

2.1 EMV Transaction Process



Step 1: Start EMV

Step 2: Auto select application

Step 3: Read Application Data

Step 4: Card Authentication

Step 5: Processing Restrictions

Step 6: Cardholder Verification

Step 7: Terminal Risk Management

Step 8: Terminal Action Analysis

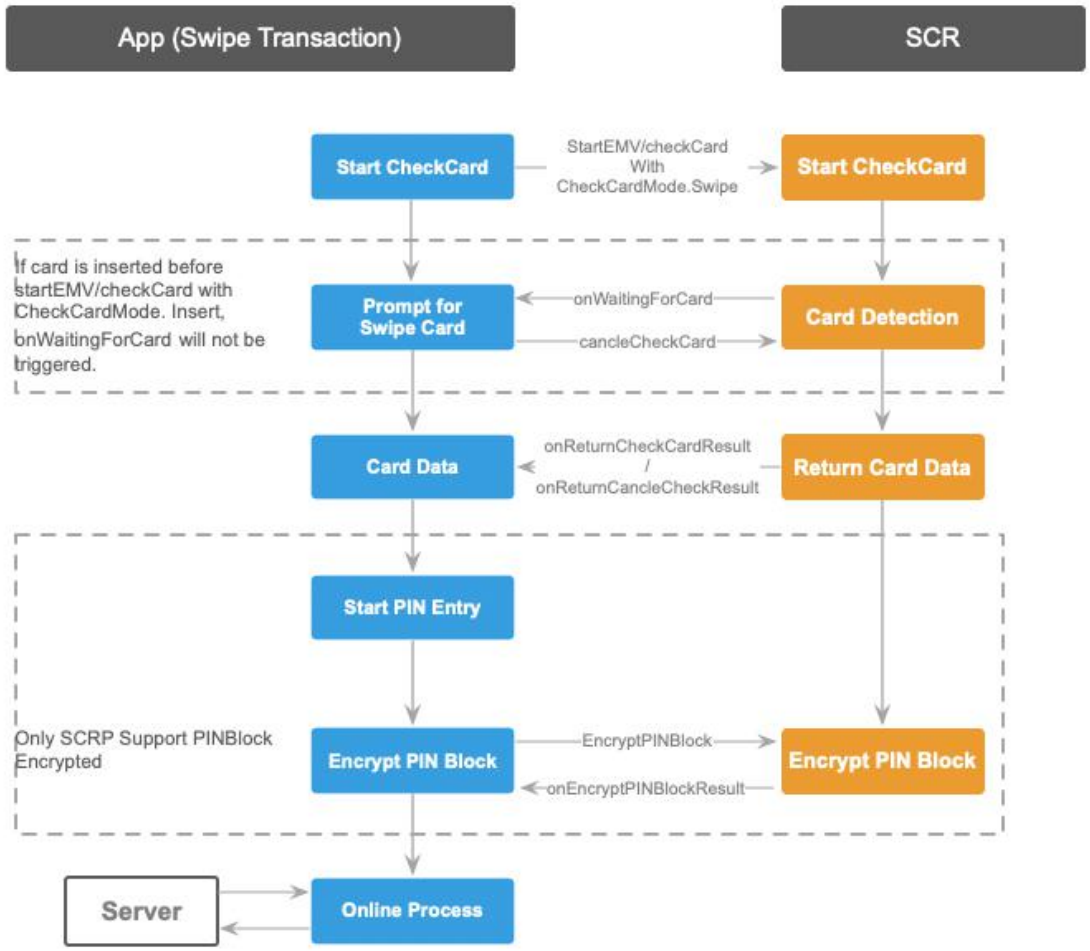
Step 9: Card Risk Management

Step 10: Online Processing

Step 11: Issuer Scripts Processing

Step 12: Completion

2.2 Magnetic Stripe Transaction



3. Development Environment

3.1 System And IDE

Android:

Android	5.1+
IDE	Android Studio or Eclipse

Language	java JDK 1.6 or above

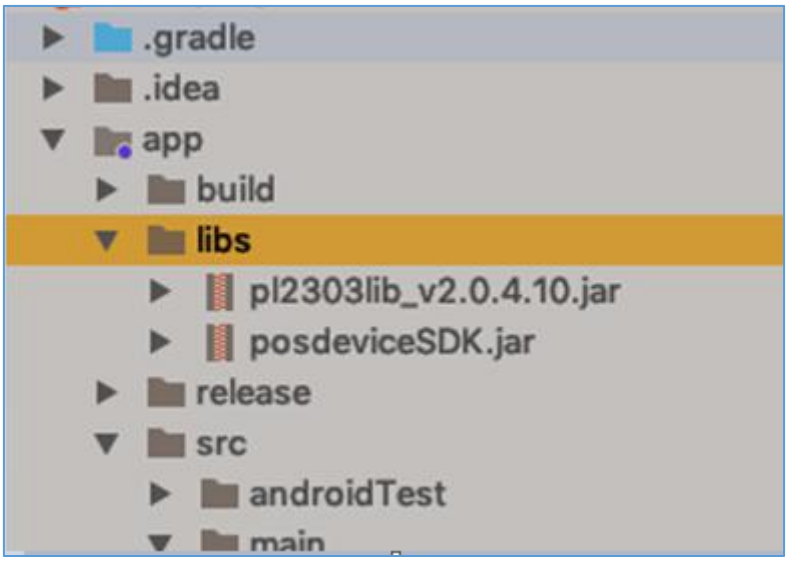
iOS:

iOS	iOS7+
IDE	Xcode
Language	Object-C or Swift

3.2 Android Studio Configure

Add *.jar, *.arr, *.so Files to project

For Android project, please import all posdeviceSDK.jar package to lib folder as shown below.

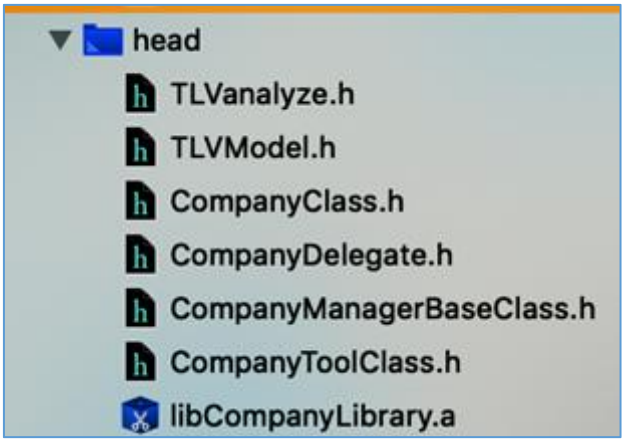


Add below repositories path into ***build.gradle*** file to enable SDK package.

```
dependencies {
    compile fileTree(include: ['*.jar'], dir: 'libs')
……
```

3.3 IOS XCode Configure

For iOS projects, please import ***libCompanyLibrary.a*** and head files to project as shown below.



3.4 Android Permission

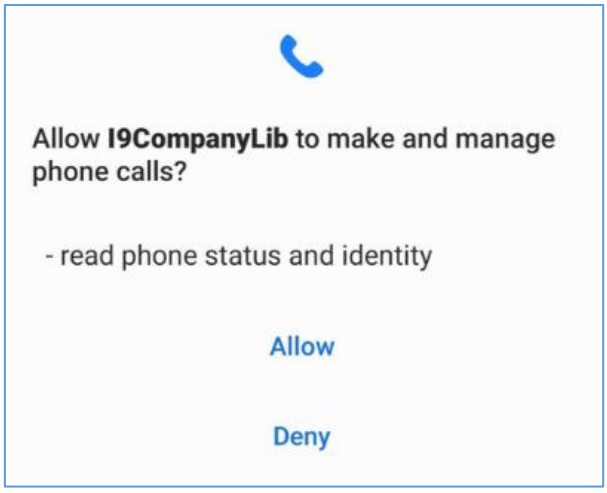
The library needs permission to use the audio and Bluetooth resource. The following lines must be added to the ***AndroidManifest.xml*** file in the APP project.

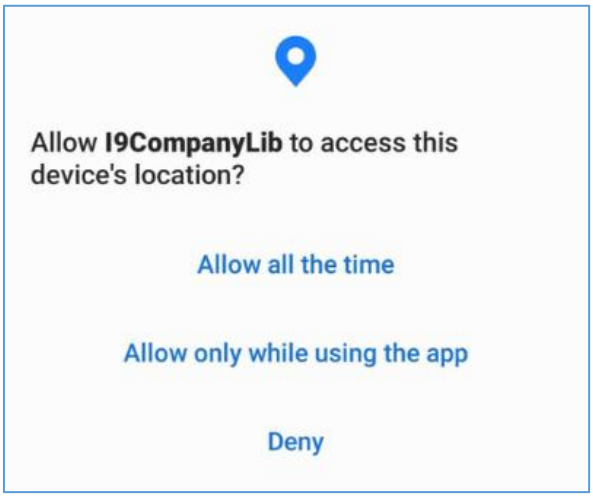
```
<uses-permission android:name="android.permission.READ_PHONE_STATE" >
</uses-permission>
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" >
</uses-permission>
<uses-permission android:name="android.permission.BLUETOOTH" >
</uses-permission>
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
```

Android 6.0+ dynamic permissions:

If the user launch the app first time, please allow these permissions below.

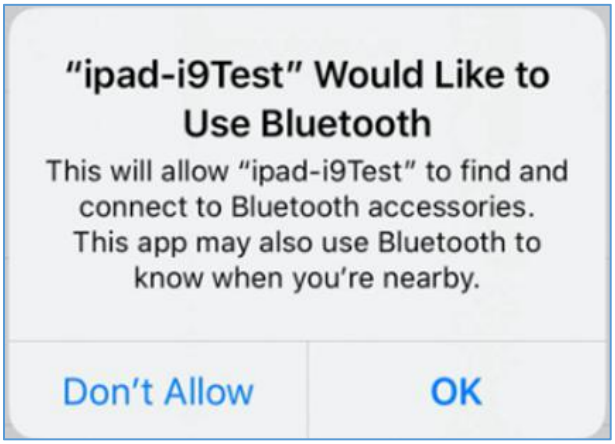
When the user logs in the app and the following authorization prompt pops up, please click "***Allow***".



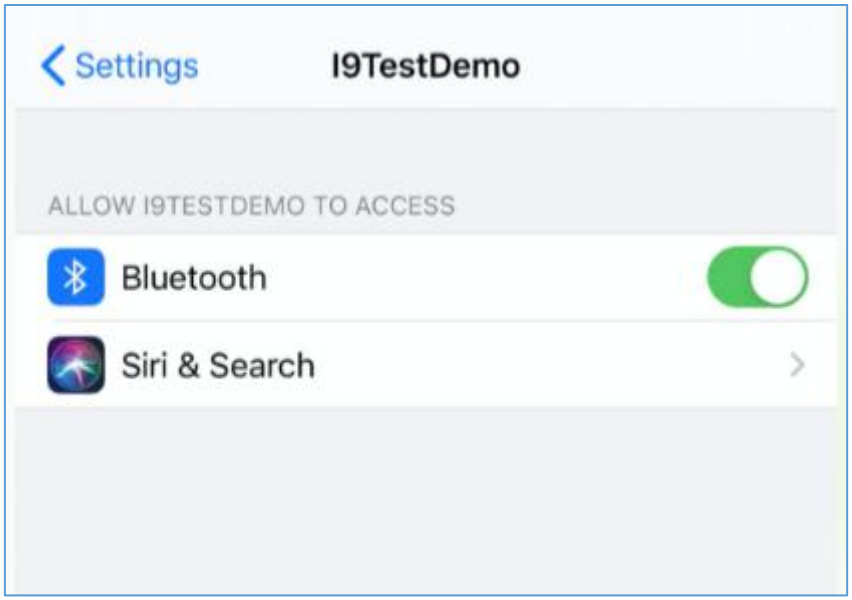


3.5 IOS Permission

Please add "Privacy - Bluetooth Peripheral Usage Description" in ***info.plist***
Authorize Bluetooth when the app is first run



IOS system ensures APP can use mobile Bluetooth
Setting/APP



3.6 Principle in SDK Methods

According to the SDK framework and the methods feature, all methods are divided into three parts:

1. Initial Devices
2. Optional Functions
3. Listener Functions

To avoid the application from blocking and improving the speed of data interaction between the smart terminal and i9, the SDK framework is designed to work under asynchronous mode.

3.7 Initialized Device

Before communicating with the device, the program needs to call the device's initialization function.

Other operations should after the Bluetooth connection.

Android :

Add below setting in Activity ***OnCreate()***

```
private BluetoothController mBluetoothController;
mBluetoothController =
BluetoothController.getInstance(I9SActivity.this,mDeviceSerListener,new
DeviceConnectionListener(){
...
}
```

IOS:

Add below setting in ***ViewController***

```
@interface I9ViewController ()<CompanyDelegate>
@property (nonatomic,strong)CompanyManagerBaseClass *mangeBase;
- (void)viewDidLoad {
[super viewDidLoad];
self.mangeBase = [CompanyManagerBaseClass getInstance];
self.mangeBase.delegate = self;
self.mangeBase.debug = YES;
self.mangeBase.deviceType = 1;
}
```

3.7.1 Scan Bluetooth Devices

Android:

function	public void scanLeDevice(final boolean enable)
Inputs	Enable true
Outputs	None
Description	Start to scan the devices
See also	The reponse method of findDevice(String mac, String name) or discoverComplete()

IOS:

Function	- (void)searchDevices:(id<CompanyDelegate>)delegate timeout:(int)timeout;
Inputs	None
Outputs	None
Description	Start to scan the devices
See also	- (void)discoverOneDevice:(CBPeripheral *)peripheral

3.7.2 Stop Scan Bluetooth Devices

Android:

Function	public void scanLeDevice(final boolean enable)
Inputs	false
Outputs	None
Description	Stop to scan the device
See also	

IOS:

Function	- (void)stopSearching;
Inputs	None
Outputs	None
Description	Stop to scan the device
See also	

3.7.3 Connect Bluetooth Device

Android:

Function	void connectDevice(String address)
Inputs	bluetooth MAC address
Outputs	None
Description	Connect to i9 by Bluetooth, using for exchanging data between APP and i9.
See also	Bluetooth connection success or fail.if success void connected() or connectFail()

IOS:

Function	- (void)openDevice:(CBPeripheral *)peripheral delegate:(id<CompanyDelegate>)cb timeout:(int)timeout;
Inputs	peripheral
Outputs	None
Description	Connect to i9 by Bluetooth, using for exchanging data between APP and i9.
See also	- (void)bleConnectState:(BOOL)isConnect{}

3.7.4 Disconnect Bluetooth Device

Android:

Function	void disconnect ()
Inputs	None
Outputs	None
Description	disconnect bluetooth device
See also	

IOS:

Function	-(void)closeDevice;
Inputs	None
Outputs	None
Description	disconnect bluetooth device
See also	

3.7.5 Release Bluetooth Resource

Android:

To avoid the port locking issue between applications when Activity accesses the hardware function, below codes must be added into Activity ***onPause()*** and ***onDestroy().***

```
if (mBluetoothController != null) {
    mBluetoothController.scanLeDevice(false);
    mBluetoothController.close();
    mBluetoothController.unRegisterReceiver();
    mBluetoothController = null;
}
```

IOS:

To avoid memory leak, below code must be added into ***ViewController viewDidDisappear***

```
if (self.mangeBase) {
[self.mangeBase closeDevice];
 self.mangeBase = nil;
}
```

3.8 Optional Functions

The Bluetooth device can be operated after the connection is done

3.8.1 Terminal Info Function

Android:

Function	void getTerminalInfo()
Inputs	None
Outputs	None

Description	Retrieve parameters about the i9device. Results are returned by getTerminalInfo. which includes: terminalNo,firmware version, bluetoothName ,bluetoothVersion,protocolType , battery level
See also	onGetTerminalInfo(TerminalInfo devInfo)

IOS:

Function	-(void)getTerminalInfo
Inputs	None
Outputs	None
Description	Retrieve parameters about the i9device. Results are returned by getTerminalInfo. which includes: terminalNo,firmware version, bluetoothName ,bluetoothVersion,protocolType , battery level,
See also	-(void)onGetTerminalInfo:(TerminalInfo *)terminalinfo status:(int)status

3.8.2 Set Terminal Automatic Shutdown Function

Android:

Function	void setAutomaticShutdown(int time)
Inputs	Time Minute unit
Outputs	None
Description	Set the terminal automatic shutdown time function. If the time is set to 0, the terminal will not automatically shutdown.
See also	onExecuteSuccess()

IOS:

Function	- (void)setAutomaticShutdown:(int)time;
Inputs	Time Minute unit
Outputs	None
Description	Set the terminal automatic shutdown time function. If the time is set to 0, the terminal will not automatically shutdown.
See also	- (void)onSetAutomaticShutdownStatus:(int)status

3.8.3 Update Terminal Time Function

Android:

Function	void setTerminalDateTime(String datetime)
Inputs	Datetime format yyyyMMdd HH:mm:ss

Outputs	None
Description	The terminal time in YYMMDDHHmmss formats should be sent in response:
See also	onExecuteSuccess()

IOS:

Function	- (void)setTerminalDateTime:(NSString *)datetime;
Inputs	Datetime format yyyyMMdd HH:mm:ss
Outputs	None
Description	The terminal time in YYMMDDHHmmss formats should be sent in response:
See also	- (void)onSetTerminalDateTime:(int)status

3.8.4 Get Terminal Time

Android:

Function	void getTerminalDateTime()
Inputs	None
Outputs	Datetime format yyMMdd HH:mm:ss
Description	The terminal time in YYMMDDHHmmss
See also	onGetTerminalDateTime(String termianlTime);

IOS:

Function	- (void)getTerminalDateTime
Inputs	None
Outputs	Datetime format yyMMdd HH:mm:ss
Description	
See also	-(void)onGetTerminalDateTime:(NSString *)datetime status:(int)status;

3.8.5 Operating Stop Function

Android:

Function	void stopTrade()
Inputs	None
Outputs	None
Description	When the terminal enters the swipe state, the stop function can be called to cancel the terminal operation.

See also	onExecuteSuccess()

IOS:

Function	- (void)stopTrade;
Inputs	None
Outputs	None
Description	When the terminal enters the swipe state, the stop function can be called to cancel the terminal operation.
See also	- (void)onStopTrade:(int)status;

3.8.6 Calculate MAC

If in Activity requires to calculate MAC, add below function into Activity for calculating MAC function.

Android:

Function	void calculateMac(byte[] data)
Inputs	MAC_ data
Outputs	None
Description	If in Activity requires to calculate MAC, add below function into Activity for calculating MAC function.
See also	onCaculateMac(String mac)

IOS:

Function	- (void)calculateMac:(int)mode data:(NSString *)data;
Inputs	MAC_mode =1, data is MAB 8 bytes
Outputs	None
Description	If in Activity requires to calculate MAC, add below function into Activity for calculating MAC function.
See also	- (void)onCalculateMac:(NSString *)mac status:(int)status

3.8.7 PIN Encrypt(Not yet supported)

The i9 device does not have the pin input function, so the user needs to enter the pin on the APP, then the APP packed PIN Block(PIN Block length is 8 bytes), and send the PIN Block to the terminal and use the PIN key encryption.

Android:

Function	void PINEntry(byte[] pinBlock)

Inputs	pinBlock 8bytes
Outputs	None
Description	User PIN encrypted
See also	onPinEncrypted(byte[] pin)

IOS:

Function	- (void)PINEntry:(NSString *)pin;
Inputs	pinBlock 8 length
Outputs	None
Description	User PIN encrypted
See also	- (void)onPINEntryResult:(NSString *)pin status:(int)status;

3.8.8 Get Card Number(Not yet supported)

Function	void getCardNo(String timeout)
Inputs	timeout
Outputs	None
Description	Get Card Number
See also	onGetCardNo(String cardNo)

3.8.9 Set Buzzer(Not yet supported)

Function	void doSetBuzzerOperation(String time)
Inputs	The buzzer ringing's time
Outputs	None
Description	Set the buzzer
See also	onExecuteSuccess()

3.8.10 Control LED Light (Not yet supported)

Function	void (String time)
Inputs	None
Outputs	None
Description	Get Tracks
See also	onExecuteSuccess()

3.8.11 Get Device Battery Capacity

Android:

Function	int getBatteryPower()
Inputs	None
Outputs	100%,75%,50%,25%,5%
Description	Get device Battery
See also	onGetBatteryPower(int batterypower);

IOS:

Function	- (void)getBatteryPower
Inputs	None
Outputs	100%,75%,50%,25%,5%
Description	Get device Battery
See also	- (void)onGetBatteryPower:(int)batteryPower status:(int)status;

3.9 EMV Functions

3.9.1 Download AID Parameters

Use this function to setup AID information. If there is no value set for AID, it will employ default value automatically.

Parameters	Description
String AID	To set AID value
int Asi	The application selected index
String AppVerNum	The application version
String TacDefault	Specifies the acquirer's conditions that cause a transaction to be rejected if it might have been approved online, but the terminal is unable to process the transaction online
String TacOnline	Specifies the acquirer's conditions that cause a transaction to be transmitted online
String TacDecline	Specifies the acquirer's conditions that cause the denial of a transaction without attempt to go online
String FloorLimit	The amount above which credit card transactions must be authorized before being processed.
String Threshold	Threshold Value for Biased Random Selection

int MaxTargetPercent	Maximum Target Percentage to be used for Biased Random Selection
int TargetPercent	Target Percentage to be Used for Random Selection
String TermDDOL	DDOL is to be used for constructing the INTERNAL AUTHENTICATE command if the DDOL in the card is not present
String vlptranslimit	The amount of the transaction limit in terminal
String termcvm_limit	The limitation of CVM
String clessofflineamt	The lowest amount of offline transaction for contactless reader.

Android:

Function	Void downloadAIDParameters(AIDParameters aIDParameters)
Inputs	AIDParameters
Outputs	None
Description	download AID Parameters
See also	onExecuteSuccess()

IOS:

Function	- (void)downloadAIDParameters:(AIDParameters *)aIDParameters;
Inputs	AIDParameters
Outputs	None
Description	download AID Parameters
See also	- (void)onDownloadAIDParameters:(int)status

3.9.2 Clear Terminal AID Parameters

Note: This command will be used when the terminal AID parameter is full. It is used to delete the terminal AID parameter. The AID needs to be updated in time or the IC card reading will fail.

Android:

Function	void clearAIDParameters(AIDParameters aIDParameters)
Inputs	AIDParameters
Outputs	None
Description	clear Terminal AID Parameters
See also	onExecuteSuccess()

IOS:

Function	- (void)clearAIDParameters;
Inputs	None
Outputs	None
Description	clear Terminal AID Parameters
See also	- (void)onClearAIDParameters:(int)status

3.9.3 Download CA Public Key

Use this function to setup CAPK information. If there is no value set for CAPK, it will use default value automatically.

Parameters	Description
String RID	To set RID value
int CAPKI	Identifies the certification authority's public key in conjunction with the RID
int HashInd	Hash Index
int ArithInd	Algorithm Index
String Modul	Value of the modulus part of the Certification Authority Public Key
String Exponent	Value of the exponent part of the CAPK
String ExpireDate	The expiration date of CAPK
String CheckSum	A check value calculated on the concatenation of all parts of CAPK

Android:

Function	void downloadPublicKey(CAPublicKey caPublicKey)
Inputs	CAPublicKey
Outputs	None
Description	download EMV CA Public Key
See also	onExecuteSuccess()

IOS:

Function	- (void)downloadPublicKey:(CAPublicKey *)caPublicKey;
Inputs	CAPublicKey
Outputs	None
Description	download EMV CA Public Key
See also	- (void)onDownloadPublicKey:(int)status

3.9.4 Clear CA Public Key

Note: This command will be used when the terminal CA public Key is full. It is used to delete the terminal CA public Key. The CA public Keyneeds to be updated in time or the IC card reading will fail.

Android:

Function	void clearPublicKey(CAPublicKey caPublicKey)
Inputs	CAPublicKey
Outputs	None
Description	clear Terminal EMV Public Key
See also	onExecuteSuccess()

IOS:

Function	- (void)clearPublicKey;
Inputs	None
Outputs	None
Description	clear Terminal EMV Public Key
See also	- (void)onClearPublicKey:(int)status

3.9.5 Start EMV Process

Use this function to start the EMV process.

Below, are the Sample code about how to get EMV TAG data.

The Limit consumption Of MasterCard NFC is 1000.

Andorid:

Function	void startEmvProcess(int timeOut,byte[] tradeData)
Inputs	timeOut,TradeData format is TLV description in 5.3
Outputs	None
Description	Check the status of the Magnetic Card Reader, the EMV Card reader, or NFC transceiver. It checks if a card has been swiped, a NFC card has been tapped or an EMV card is inserted. The result is returned by the onReadCarddelegate method.
See also	onReadCard()

EMV data event:

```
@Override
public void onReadCard(CardInfo cardInfo) {
    StringBuffer sb = new StringBuffer();
    sb.append("onReadCard");
    sb.append("\ncontrolModel: "+cardInfo.getControlModel());
    sb.append("\ncardType: "+cardInfo.getCardType());
    sb.append("\ncardexpiryDate: "+cardInfo.getCardexpiryDate());
    sb.append("\ntracks: "+Util.BinToHex(cardInfo.getTracks()));
    sb.append("\ncardNo: "+cardInfo.getCardNo());
    sb.append("\nPAN: "+cardInfo.getPAN());
    sb.append("\nCardSerial: "+cardInfo.getCardSerial());
    sb.append("\nCVM: "+cardInfo.getCVM());
    sb.append("\nicTracks: "+ Util.BinToHex(cardInfo.getIcTracks()));
    sb.append("\nterimalNo: "+cardInfo.getTerimalNo());
    if(cardInfo.getNfcTradeRespose()>0){
        sb.append("\nNfcTradeRespose: "+cardInfo.getNfcTradeRespose());
    }
    logger.error(sb.toString());
    mHandler.sendMessage(mHandler.obtainMessage(0,sb.toString()));
}
```

IOS:

Function	- (void)startEmvProcess:(int)timeout tradeData:(TradeData*)tradeData;
Inputs	timeOut,TradeData
Outputs	None
Description	Check the status of the Magnetic Card Reader, the EMV Card reader, or NFC transceiver. It checks if a card has been swiped, a NFC card has been tapped or an EMV card is inserted. The result is returned by the onReadCarddelegate method.
See also	- (void)onReadCard:(CardInfo *)cardInfo status:(int)status

3.9.6 EMV Online Process

Android:

Function	void sendOnlineProcessResult(byte[] icData)
Inputs	icData Tlv:type + length + value
Outputs	None

Description	Send transaction results from the processor back to EMVSwipe
See also	onRequestOnlineProcess()

IOS:

Function	- (void)sendOnlineProcessResult:(NSString*)data;
Inputs	cData Tlv:type + length + value
Outputs	None
Description	Send transaction results from the processor back to EMVSwipe
See also	- (void)onSendOnlineProcessResult:(int)onlineResult scriptResult:(NSString *)scriptResult data:(NSString *)data status:(int)status;

3.10 APDU Transparent Transfer Functions

Sometimes we need a requirement to send APDU data directly to the card. The i9 SDK provides such a method.

The specific steps are:

1. Open Transparent Transfer Function
2. Send APDU Packet
3. Close Transparent Transfer Function

3.10.1 Open APDU Transparent Transfer Function

Android:

Function	void powerOnAPDU(int type, int timeout)
Inputs	type 0 is IC , 1 is NFC timeout
Outputs	None
Description	Turn on the NFC or IC transceiver
See also	onPowerOnAPDU(int cardType, byte[] data);

IOS:

Function	- (void)powerOnAPDU:(int)type timeout:(int)timeout
Inputs	type 0 is IC , 1 is NFC timeout
Outputs	None
Description	Turn on the NFC or IC transceiver

See also	- (void)onPowerOnAPDU:(int)type cardData:(NSString *)data status:(int)status;

3.10.2 Send APDU Packet

Android:

Function	void sendApdu(int type, List<byte[]> listbytes, int timeout)
Inputs	type 0 is IC , 1 is NFC listbytes APDU data list {command1, command2,command3....} timeout
Outputs	None
Description	Send data to EMV card in raw APDU formats by nfc or IC
See also	onAPDUResponsedata()

IOS:

Function	- (void)sendApdu:(int)type apdu:(NSArray*)apduData timeout:(int)timeout
Inputs	type 0 is IC , 1 is NFC listbytes APDU data list {command1, command2,command3....} timeout
Outputs	None
Description	Send data to EMV card in raw APDU formats by nfc or IC
See also	- (void)onSendApdu :(NSArray *)data status:(int)status;

3.10.3 Close APDU Transparent Transfer Function

Android:

Function	void powerOffAPDU()
Inputs	None
Outputs	None
Description	Stop APDU operating
See also	onExecuteSuccess()

IOS:

Function	- (void)powerOffAPDU
Inputs	None
Outputs	None

Description	Stop APDU operating
See also	- (void)onPowerOffAPDU:(int)status;

3.10 Key Loading Functions

Some keys was hardcoded in debug firmware please refer to ***1.3 Firmware Hardcode Transaction Keys***

3.11.1 Master Key / Fixed Key /DUKPT IPEK Injection

SCR devices should inject mater key, fixed key and DUKPT IPEK by Host Secure Module in factory secure room before shipment.

3.11.2 Download Session Key

TR-31 Key Loading Interface (Not yet supported)

Android:

Function	void downloadWorkKey(int mkIndex, byte[] PINkey,byte[] MACkey, byte[] DESkey)
Inputs	mkIndex PINkey MACkey DESkey
Outputs	None
Description	update
See also	onExecuteSuccess()

IOS:

Function	- (void)downloadWorkKey:(int)index PINkey:(NSString *)PINkey MACkey:(NSString *)MACkey DESkey:(NSString *)DESkey
Inputs	index PINkey MACkey DESkey
Outputs	None
Description	update
See also	- (void)onDownloadWorkKey:(int)status

4. Listener Functions

4.1 onGetTerminalInfo

Android:

Function	void onGetTerminalInfo(TerminalInfo terminalInfo)
Inputs	TerminalInfo TerminalType,.TerminalNo, SoftVersion),SoftVersionDate,BluetoothName,BluetoothMAC ,BluetoothVersion,ProtocolType,MerchaantNo.
Outputs	None
Description	Callback of pos info response
See also	

IOS:

Function	- (void)onGetTerminalInfo:(TerminalInfo *)terminalinfo status:(int)status
Inputs	Terminalinfo: the class of terminalinfo Status: the code of terminal return,if 0,is success
Outputs	None
Description	callback of getTerminal
See also	

4.2 onGetCardNo

Function	void onGetCardNo(String cardNo)
Inputs	Card No
Outputs	None
Description	Callback of pos info response
See also	getCardNo()

4.3 onCaculateMac

Android:

Function	void onCaculateMac(String mac)
Inputs	MAC
Outputs	None
Description	Callback of pos info response

See also	CaculateMac()

IOS:

Function	- (void)onCalculateMac:(NSString *)mac status:(int)status
Inputs	Mac: data of mac Status: the code of terminal return,if 0,is success
Outputs	None
Description	callback of calculateMac
See also	- (void)calculateMac:(int)mode data:(NSString *)data;

4.4 onPinEncrypted(Not yet supported)

Android:

Function	void onPINEntryResult(String pin)
Inputs	PIN
Outputs	None
Description	Callback of pos info response
See also	

IOS:

Function	- (void)onPINEntryResult:(NSString *)pin status:(int)status
Inputs	Pin: encrypt of pin Status: the code of terminal return,if 0,is success
Outputs	None
Description	callback of pinEncrypt
See also	

4.5 onTimeOut

Android:

Function	void onTimeOut()
Inputs	None
Outputs	None
Description	timeOut
See also	

IOS:

Function	- (void)sendCommandTimeout
Inputs	None
Outputs	None
Description	timeOut
See also	

4.6 onError

Android:

Function	void onError(int code, String msg)
Inputs	Error Code ,error MSG
Outputs	None
Description	SDK error information response
See also	

IOS:

Function	- (void)onError:(NSInteger)code msg:(NSString *)msg
Inputs	code 1 blueTooth disconnect 2 BlueToothStatePoweredOff 3 blueTooth receive data error
Outputs	None
Description	SDK error information response
See also	

4.7 onWaitingcard

Android:

Function	void onWaitingcard()
Inputs	None
Outputs	None
Description	In the callback to waiting user operating
See also	

IOS:

Function	- (void)onWaitingcard
Inputs	None
Outputs	None

Description	terminal can swipe
See also	

4.8 onWaitingPIN

Only Mpos support

Android:

Function	void onWaitingPIN()
Inputs	None
Outputs	None
Description	In the callback to waiting user Pin operating
See also	

IOS:

Function	- (void)onWaitingPIN
Inputs	None
Outputs	None
Description	In the callback to waiting user Pin operating
See also	

4.9 onICWaitingOper

Android:

Function	void onICWaitingOper()
Inputs	None
Outputs	None
Description	IC card have inserted
See also	

IOS:

Function	- (void)onICCardInsertion
Inputs	None
Outputs	None
Description	IC card have inserted
See also	

4.10 **NFC Card Detection**

Android:

Function	void onNFCCardDetection();
Inputs	None
Outputs	None
Description	NFC card detected
See also	

IOS:

Function	- (void)onNFCCardDetection
Inputs	None
Outputs	None
Description	NFC card detected
See also	

4.10 onReadCard

Android:

Function	void onReadCard(CardInfo cardInfo);
Inputs	CardInfo private String cardNmae; private String cardNo; private int cardType; private String cardexpiryDate; private String PAN; private String CardSerial; private String CVM; private String EMVData; private byte[] tracks; private byte[] icTracks; private byte[] track1; private byte[] track2; private byte[] track3; private byte[] PIN; private String MAC; private String terimalNo; private int nfcTradeRespose; private int NFCCardType;
Outputs	None

Description	Callback of track data response
See also	

IOS:

Function	- (void)onReadCard:(CardInfo *)cardInfo status:(int)status
Inputs	CardInfo @property (nonatomic,copy)NSString *KSN; @property (nonatomic,copy)NSString *SN; @property (nonatomic,copy)NSString *ControlModel; @property (nonatomic,copy)NSString *psamNo; @property (nonatomic,copy)NSString *cardName; @property (nonatomic,copy)NSString *cardNo; @property (nonatomic,assign)int cardType; @property (nonatomic,copy)NSString *cardexpiryDate; @property (nonatomic,copy)NSString *PAN; @property (nonatomic,copy)NSString *cardSerial; @property (nonatomic,copy)NSString *CVM; @property (nonatomic,copy)NSString *tracks; @property (nonatomic,copy)NSString *trackLen; @property (nonatomic,copy)NSString *encryTrackLen; @property (nonatomic,copy)NSString *PIN; @property (nonatomic,copy)NSString *MAC; @property (nonatomic,copy)NSString *TUSN; @property (nonatomic,copy)NSString *icdata; @property (nonatomic,copy)NSString *random; @property (nonatomic,copy)NSString *result; @property (nonatomic,copy)NSString *deninalReason; @property (nonatomic,copy)NSString *kernelType; @property (nonatomic,copy)NSString *outcomeParameterSet; @property (nonatomic,copy)NSString *userInterfaceRequestData; @property (nonatomic,copy)NSString *errorIndication; @property (nonatomic,assign)int NFCCardType;
Outputs	None
Description	Callback of track data response
See also	

4.11 onStopTradeSuccess

Android:

Function	void onStopTradeSuccess();

Inputs	Error Code ,error MSG
Outputs	None
Description	None
See also	

IOS:

Function	- (void)onStopTrade:(int)status
Inputs	Status: the code of terminal return,if 0,is success
Outputs	None
Description	callback of stopTrade
See also	

4.12 onExecuteSuccess

Android:

Function	void onExecuteSuccess(int code)
Inputs	Success Code
Outputs	None
Description	
See also	

4.13 onAPDUResponsedata

Android:

Function	void onSendApdu(List<byte[]> lists)
Inputs	Apdu List<byte[]>
Outputs	None
Description	Response of batch APDU instruction execution
See also	

IOS:

Function	- (void)onSendApdu:(NSArray *)data status:(int)status;
Inputs	Data: apdu result Status: the code of terminal return,if 0,is success
Outputs	None
Description	callback of sendApdu
See also	

4.14 onBatteryPower

Android:

Function	void onGetBatteryPower(int batterypower);
Inputs	Battery power
Outputs	None
Description	Response of batch APDU instruction execution
See also	

IOS:

Function	- (void)onGetBatteryPower:(int)batteryPower status:(int)status
Inputs	batteryPower: batteryPower Status: the code of terminal return,if 0,is success
Outputs	None
Description	callback of getBatteryPower
See also	

4.15 onTerminalDateTime

Android:

Function	onGetTerminalDateTime(String termianlTime)
Inputs	datetime
Outputs	None
Description	TerminalDateTime
See also	

IOS:

Function	-(void)onGetTerminalDateTime:(NSString*)datetime status:(int)status;
Inputs	datetime
Outputs	None
Description	TerminalDateTime
See also	

5. Constant Class and Enum

5.1 Customer(enum)

Not currently supported, please refer to Demo

5.2 ErrorCode(classs)

Error Code	*Usage*
ERROR_CODE_201	Received data length error
ERROR_CODE_202	Received data check error
ERROR_CODE_203	Received data parse error
ERROR_CODE_204	Received data error, means the last command has not been executed successful
ERROR_CODE_205	Reserve，not useful for the current released SDK version
ERROR_CODE_206	Reserve，not useful for the current released SDK version
ERROR_CODE_207	Reserve，not useful for the current released SDK version
ERROR_CODE_208	Reserve，not useful for the current released SDK version

5.3 TradeTag(class)

TAG	*Usage*
TAG_1F01	Card model
TAG_1F02	Control identifier
TAG_1F03	Date and time
TAG_1F04	Key Type
TAG_1F07	Random data
TAG_1F08	Amount
TAG_1F09	Additional transition information
TAG_1F0A	Additional transition data
TAG_1F0B	Callback tip showed on the screen of pos, but i9 is not supported
TAG_1F74	calculate data, ex:calculate mac、calculate pin
TAG_DF21	Reserve，not useful for the current released SDK version
TAG_DF22	Reserve，not useful for the current released SDK version

6. Error Code

Error Code Table:

Return Code(HEX)	*Description*
0x00	Executed command successfully
0x01	Command execution timeout
0x0A	User exits
0x70	External FLASH error
0x72	Track data error
0x73	Card expiration date failed
0x74	EMV data error
0x80	Data received correctly
0x81	Track analysis error
0x82	Failure to swipe card
0x83	Correct card resolution
0x84	Insert IC card
0x85	IC card pullout
0x86	IC Card reading failure
0x89	Please remove the NFC card
0x8A	Select NFC card
0x8B	Failed to read NFC card
0x8C	Please trade by other methods.
0x8D	No service
0x8E	Card refuse to trade
0x8F	Show NFC card
0x90	Failure of IC Card reset
0x94	TLS hand shake exception
0x9A	Activation failure
0x9B	Session key error
0x9C	Verification signature error
0x9D	Signature error
0x9E	Overload
0x9F	Terminal not self-destructed
0xD4	NFC card error
0xD7	Please show one card
0xEA	Terminal locking
0xF1	Unrecognized master command code
0xF2	Unrecognized subcommand code
0xF3	Instruction not support this version.

0xF4	Random number length error
0xF5	Unsupported components
0xF6	Unsupported patterns
0xF7	Data length error
0xFC	Data parameter error
0xFD	Terminal ID error
0xFF	Data CRC error