Dspread Technology (Beijing) Inc

QPOS SDK
Integration Guide

Document Version：<7.7.1>
Last Updated: 2025.10.21

Catalogue

Document Overview

This document serves as the official integration guide for the QPOS Android SDK. It provides **detailed, step-by-step instructions** for developers to connect QPOS payment terminal devices (including QPOS, QPOS Reader, and SPOS) to Android applications.
The content covers core topics such as system requirements, SDK API usage, transaction processes, and common operation guidelines. The goal is to help developers quickly complete integration and ensure the payment function is **secure and stable**.

1. Version History

Version	Update Date	Author	Key Modifications
1.0.0	2013.07.12	Longhui.xiao	Initial release (Chinese Version)
2.0.0	2013.08.10	William.tang	Released English Version
2.1.0	2013.09.20	William.tang	Added support for Audio and Bluetooth communication modes
2.2.0	2013.09.21	Xu.Wang	Included QPOS Reader Series-related content
2.3.0	2014.03.11	Longhui Xiao	Added ICC APDU operation interface
2.4.0	2014.03.26	Longhui Xiao	Added interface for setting device sleep time
2.5.0	2014.04.08	Longhui	Added interface for updating work keys

Dspread Technology (Beijing) Inc

Version	Update Date	Author	Key Modifications
		Xiao	
2.6.0	2015.02.11	Yanhao Lu	Optimized document structure and content description
2.7.0	2015.09.07	Longhui Xiao	Added resetPosStatus interface; further refined document details
2.8.0	2016.12.27	Qianmeng Chen	Added doSetManagementKey and doSetBuzzerOperation interfaces
2.9.0	2017.01.04	Qianmeng Chen	Added doUpdateIPEKOperation interface; updated document content
3.0.0	2017.01.11	Qianmeng Chen	Added updateEmvAPP and updateEmvCAPK interfaces
3.1.0	2017.03.03	Qianmeng Chen	Added scanQpos2Mode and stopQpos2Mode interfaces
3.2.0	2022.04.25	Qianmeng Chen	Added APIs: operateLEDByType and playBuzzerByType
3.3.0	2022.09.19	Shasha Lv	Revised descriptions of onRequestGenerateTransportKey and onReturnUpdateIPEKResult
3.4.0	2024.01.04	Shasha Lv	Added APIs: sendCvmPin and getEncryptData()
3.5.0	2024.01.05	Shasha Lv	Added APIs related to online files
3.6.0	2025.05.21	Shasha Lv	Modified initialization methods and listener methods
7.7.0	2025.09.19	Qianmeng Chen	Added Smart POS connection method and PIN input function. Optimise the logic of different modules in the document
7.7.1	2025.10.21	Mengqiu Ma	Supplement the API and adjust the document style

2. Preface

2.1 Purpose & Scope

Purpose: This document standardizes the integration process of QPOS Android SDK, clarifies the usage of SDK APIs, transaction logic, and troubleshooting methods, and ensures that developers can correctly connect QPOS devices to Android applications to achieve secure and stable payment transactions.

Dspread Technology (Beijing) Inc

Scope: Covers QPOS series devices (QPOS, QPOS Reader, SPOS) and their Android SDK (Version 7.7.0). It includes system environment requirements, SDK framework, permission configuration, API parameter descriptions, transaction flow diagrams, and delegate method callbacks, but does not involve hardware maintenance of QPOS devices.

2.2 Target Audience

- Android application developers responsible for integrating mobile payment functions.
- Technical support engineers who provide QPOS SDK integration guidance.
- Testers who verify the compatibility and stability of QPOS device connections.

3 Glossary & Definitions

Term	Full Name / Definition
SDK	Software Development Kit (a toolset for integrating QPOS devices with applications)
QPOS	Mobile payment card reader with PIN pad (supports multiple communication modes)
QPOS Reader	Mobile payment card reader without PIN pad (supports Audio/USB communication)
SPOS	Mobile payment card reader with PIN pad and touch screen (supports signature capture)
EMV	Europay, MasterCard, Visa (a global standard for chip card payments)
ICC	Integrated Circuit Card (commonly known as "chip card")
APDU	Application Protocol Data Unit (data interaction protocol between card and terminal)
DUKPT	Derived Unique Key Per Transaction (a key management standard for secure transactions)
KSN	Key Serial Number (unique identifier for key derivation)
IPEK	Initial PIN Encryption Key (initial key for PIN encryption)
TLV	Tag-Length-Value (a data encoding format used in EMV transactions)

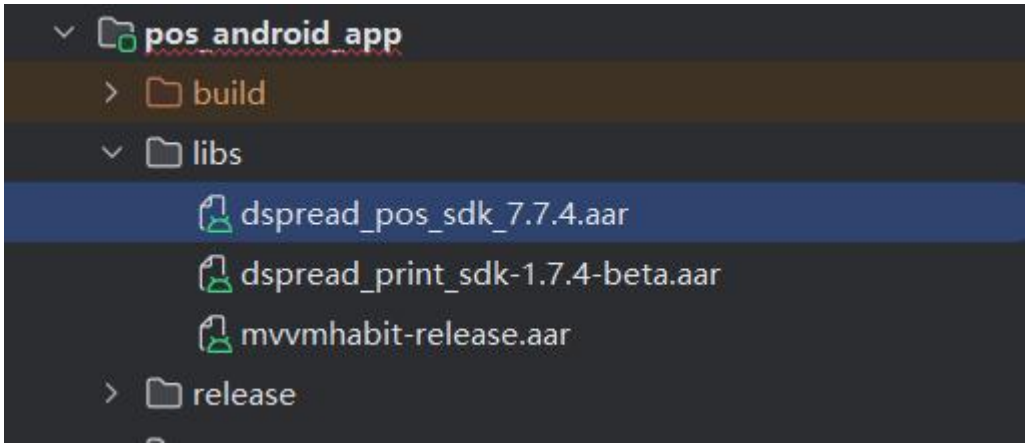
4. SDK Overview

This article is the integration development guide document for the Dspread unified SDK standard. It is intended to guide users on how to use the SDK, assuming that the reader is already familiar with the basic usage of the IDE (Android Studio) and has a certain foundation in Android programming.

4.1 Implement SDK Method

Follow these steps to integrate the SDK into your Android project:

1. **Download the SDK AAR Package**: Obtain the SDK AAR package from the designated source (SDK aar link)
2. **Place the AAR Package in the Project**: Copy the downloaded AAR package (e.g., dspread_pos_sdk_7.7.4.aar) into the **libs** folder of your Android project.



3. **Configure the App-Level Build File**: Add the following code to the app/build.gradle (or app/build.gradle.kts) file to import the AAR package:.

```
implementation files('libs/dspread_pos_sdk_7.7.4.aar')
```

4.2 Supported System Versions & IDE

- Currently, the SDK supports **API-19** (Android 4.4)Currently
- The SDK only supports integration with **Android Studio** and **IntelliJ**.

4.3 Android permission

The QPOS SDK requires specific permissions to access Bluetooth and location resources (for Bluetooth scanning). Add the following permissions to the **AndroidManifest.xml** file of your application:

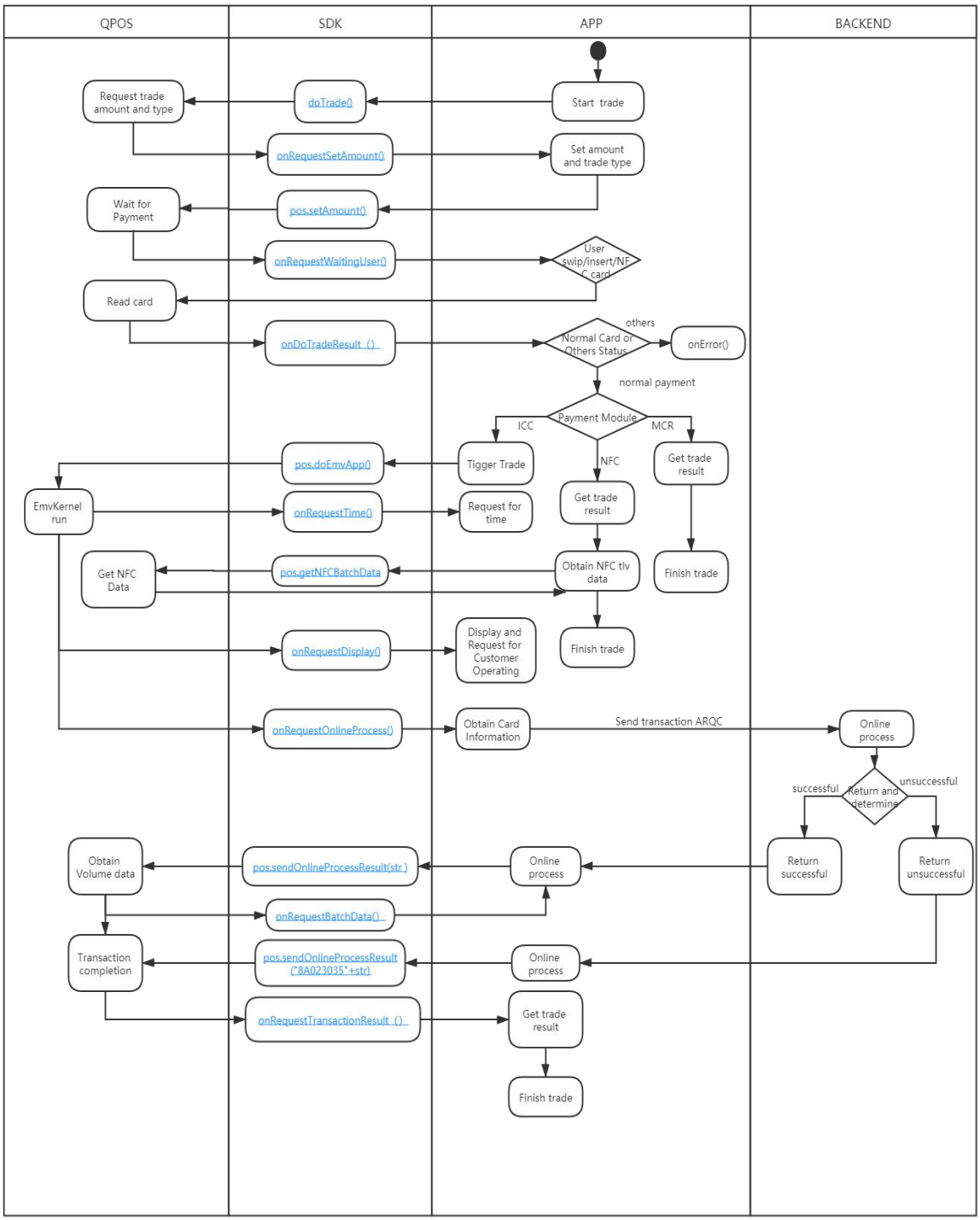
```
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />

<!-- Required permissions for Android 12 and above versions -->
<uses-permission
        android:name="android.permission.BLUETOOTH_SCAN"
        android:usesPermissionFlags="neverForLocation"
        tools:targetApi="s" />
<uses-permission android:name="android.permission.BLUETOOTH_CONNECT" />
<uses-permission android:name="android.permission.BLUETOOTH_ADVERTISE" />

<!-- Location permissions -->
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
```

Note: For Android 6.0 (API Level 23) and above, you need to request runtime permissions for dangerous permissions (e.g., ACCESS_FINE_LOCATION, BLUETOOTH_CONNECT). Do not rely solely on manifest declarations.

4.4 API Reference

This section details the core APIs provided by the QPOS SDK, categorized by function.

Before reviewing the API specifications, please refer to the transaction flow chart below to better understand the process for app implementation.



4.4.1 Connection APIs

These APIs are used to initialize, connect, and disconnect QPOS devices (supports UART, Bluetooth, BLE, and USB communication modes).

- **Init QPOSService**

API Methods	Description
getInstance(Context ctx, CommunicationMode mode)	Get the QPOSService instance by initializing mode
initListener(QPOSServiceListener listener)	Initializes the QPOSServiceListener callback; Since the handler is not initialized in the SDK, the app must enable UI threads to perform any UI operations within the callback.
initListener(Handler handler, QPOSServiceListener listener)	Initializes the QPOSServiceListener callback; In this version, the handler is initialized within the SDK, allowing the app to perform UI operations directly inside the callback.

- **UART connection**

API Methods	Description
openUart()	Open the SPOS devices
closeUart()	Close UART devices

- **Bluetooth connection**

API Methods	Description
scanQPos2Mode(Context context, long time)	Scan bluetooth devices
stopScanQPos2Mode()	Stop scan bluetooth devices
connectBluetoothDevice(boolean auto, int bondtime, String blueToothAddress)	Connect normal bluetooth devices
disconnectBT()	Disconnect normal bluetooth device

- **Bluetooth BLE connection**

API Methods	Description
startScanQposBLE(final int timeout)	Scan BLE devices
stopScanQposBLE()	Stop scan BLE devices
connectBLE(String blueToothAddress)	Connect BLE devices
disconnectBLE()	Disconnect BLE device

- **USB connection**

API Methods	Description
openUsb(UsbDevice usbDevice)	Open USB
closeUsb()	Close USB

Detailed API Specifications (Connection APIs)

Below is the detailed parameter, output, and usage description for each connection API:

getInstance

Item	Details
Signature	QPOSService getInstance(Context ctx, CommunicationMode mode)
Inputs	**Ctx**: context **Mode**: CommunicationMode.UART/BLUETOOTH/BLUETOOTH_BLE/USB_OTG_CDC_ACM
Outputs	Return the QPOSService instance
Usage Example	QPOSService pos = QPOSService.getInstance(context, CommunicationMode.UART)

initListener

Item	Details
Signature	void initListener(QPOSServiceListener listener)
Inputs	**listener**: An instance of QPOSServiceListener
Outputs	None

initListener

Item	Details
Signature	void initListener(Handler handler, QPOSServiceListener listener)
Inputs	**handler**: The handler to perform UI operation **listener**: An instance of QPOSServiceListener
Outputs	None

openUart

Item	Details
Signature	void openUart()
Inputs	None
Outputs	None

closeUart

Item	Details
Signature	void closeUart()
Inputs	None
Outputs	None
Callback	onRequestQposDisconnected

scanQPos2Mode

Item	Details
Signature	boolean scanQPos2Mode(Context context, long time)
Inputs	context: context instance time: the timeout for scan device
Outputs	None

stopScanQPos2Mode

Item	Details
Signature	Void stopScanQPos2Mode()
Inputs	None
Outputs	None

connectBluetoothDevice

Item	Details

Item	Details
Signature	boolean connectBluetoothDevice(boolean auto, int bondtime, String blueToothAddress)
Inputs	**auto**: if is true,can pair the device automatically, else is false, need pair the device manually **bondtime**: the bluetooth bond time **blueToothAddress**: device bluetooth mac address
Outputs	None
Callback	onRequestQposConnected

disconnectBT

Item	Details
Signature	void disconnectBT()
Inputs	None
Outputs	None
Callback	onRequestQposDisconnected

startScanQposBLE

Item	Details
Signature	void startScanQposBLE(nt timeout)
Inputs	**timeout**: the timeout for scan device
Outputs	None

stopScanQposBLE

Item	Details
Signature	void stopScanQposBLE()
Inputs	None
Outputs	None

connectBLE

Item	Details
Signature	boolean connectBLE(String blueToothAddress)
Inputs	**blueToothAddress**: device bluetooth mac address
Outputs	None
Callback	onRequestQposConnected

disconnectBLE

Item	Details
Signature	void disconnectBLE()
Inputs	None
Outputs	None
Callback	onRequestQposDisconnected

openUsb

Item	Details
Signature	void openUsb()
Inputs	None
Outputs	None
Callback	onRequestQposConnected

closeUsb

Item	Details
Signature	void closeUsb()
Inputs	None
Outputs	None
Callback	onRequestQposDisconnected

4.4.2 Transaction APIs

Provides relevant APIs for executing transactions and details the involved transaction processes. Below is a specific list of APIs:

API Methods	Description
setCardTradeMode(CardTradeMode cardTradeMode)	Set the card reading mode allowed by the device
doTrade(int timeout)	Send the command as swiping/inserting/tapping card to POS.
setAmount(String amount, String cashbackAmount, String currencyCode, TransactionType transactionType)	Set the amount and transaction type required for EMV transaction
cancelSetAmount()	Cancel the process about setting amount
sendTime(String terminalTime)	Send current time.
doEmvApp(EmvOption emvOption)	Send the command as executing the EMV transaction flow to POS
selectEmvApp(int index)	Select one application of the application list returned from EMV kernel, then set the application ID to EMV kernel
cancelSelectEmvApp()	Cancel the process about setting the application
sendPin(byte[] pin)	Input the pin to CR100
sendCvmPin(String pin, boolean isEncrypted)	Input the cipher pinblock on the client app side to CR100
cancelPin()	Set cancel input pin to CR100
pinMapSync(String datas, int timeout)	Send the pin keyboard datas to Smart Device for pin input
sendOnlineProcessResult(String tlv)	Send the received online processing result to POS
getNFCBatchData()	Obtain the tlv data of NFC

setCardTradeMode

Item	Details
Signature	void setCardTradeMode(CardTradeMode cardTradeMode)

Item	Details
Inputs	**cardTradeMode**: CardTradeMode .ONLY_INSERT_CARD, ONLY_SWIPE_CARD, TAP_INSERT_CARD, TAP_INSERT_CARD_NOTUP, SWIPE_TAP_INSERT_CARD, UNALLOWED_LOW_TRADE, SWIPE_INSERT_CARD, SWIPE_TAP_INSERT_CARD_UNALLOWED_LOW_TRADE, SWIPE_TAP_INSERT_CARD_NOTUP_UNALLOWED_LOW_TRADE, ONLY_TAP_CARD,ONLY_TAP_CARD_QF, SWIPE_TAP_INSERT_CARD_NOTUP, SWIPE_TAP_INSERT_CARD_DOWN, SWIPE_INSERT_CARD_UNALLOWED_LOW_TRADE, SWIPE_TAP_INSERT_CARD_UNALLOWED_LOW_TRADE_NEW, ONLY_INSERT_CARD_NOPIN, SWIPE_TAP_INSERT_CARD_NOTUP_DELAY
Outputs	None
Usage Example	pos.setCardTradeMode(CardTradeMode.SWIPE_TAP_INSERT_CARD);

doTrade

Item	Details
Signature	void doTrade(int timeout)
Inputs	**timeout**: timeout of trade
Outputs	None
Callback	onDoTradeResult
Usage Example	pos.doTrade(60);

setAmount

Item	Details
Signature	void setAmount(String amount, String cashbackAmount, String currencyCode, TransactionType transactionType)

Item	Details
Inputs	**amount**: how much money in cents **cashbackAmount**: cashback amount in cents **currencyCode**: currency code, like RMB is 156 **transactionType**: TransactionType.GOODS, SERVICES, CASH, CASHBACK, PURCHASE_REFUND, INQUIRY, TRANSFER, ADMIN, CASHDEPOSIT, PAYMENT, PBOCLOG, SALE, PREAUTH, ECQ_DESIGNATED_LOAD, ECQ_UNDESIGNATED_LOAD, ECQ_CASH_LOAD, ECQ_CASH_LOAD_VOID, ECQ_INQUIRE_LOG, REFUND, UPDATE_PIN, SALES_NEW, NON_LEGACY_MONEY_ADD, LEGACY_MONEY_ADD, BALANCE_UPDATE, MINI_STATEMENT, CUSTOMIZE, BALANCE
Outputs	None
Callback	onRequestSetAmount
Usage Example	pos.setAmount(“1000”, “”, "156",TransactionType.GOODS);

cancelSetAmount

Item	Details
Signature	void cancelSetAmount()
Inputs	None
Outputs	None
Callback	onRequestSetAmount
Usage Example	pos.cancelSetAmount();

sendTime

Item	Details
Signature	void sendTime(String terminalTime)
Inputs	**terminalTime**: current trade time (format: yyyyMMddHHmmss)
Outputs	None
Callback	onRequestTime
Usage Example	pos.sendTime("20210228192610");

doEmvApp

Item	Details
Signature	void doEmvApp(EmvOption emvOption)
Inputs	**emvOption**: EmvOption.START,START_WITH_FORCE_ONLINE,START_WITH_FORCE_PIN, START_WITH_FORCE_ONLINE_FORCE_PIN, START_WITH_RETURN_ICC_CARD_NUMBER, START_WITH_FORCE_ONLINE_RETURN_ICC_CARD_NUMBER, START_WITH_FORCE_PIN_RETURN_ICC_CARD_NUMBER, START_WITH_FORCE_ONLINE_FORCE_PIN_RETURN_ICC_CARD_NUMBER,
Outputs	None
Callback	None
Usage Example	pos.doEmvApp(EmvOption.START);

selectEmvApp

Item	Details
Signature	void selectEmvApp(int index)
Inputs	**index**: selectEmvApp
Outputs	None
Callback	onRequestSelectEmvApp
Usage Example	pos.selectEmvApp(1);

cancelSelectEmvApp

Item	Details
Signature	void cancelSelectEmvApp()
Inputs	None
Outputs	None
Callback	onRequestSelectEmvApp

Item	Details
Usage Example	pos.cancelSelectEmvApp();

sendPin

Item	Details
Signature	void sendPin(byte[] pin)
Inputs	**pin**: pin code length just allow 4-12
Outputs	None
Callback	onRequestSetPin
Usage Example	pos.sendPin("123456".getBytes());

sendCvmPin

Item	Details
Signature	void sendCvmPin(String pin, boolean isEncrypted)
Inputs	**pin**: this pinblock should be the ISO format-4 pinblock which meets the SPOC protocol **isEncrypted**: whether the pin is encrypted
Outputs	None
Callback	onRequestSetPin

Usage Example	String pinBlock = buildCvmPinBlock(pos.getEncryptData(), newPin);// build the ISO format4 pin block
	pos.sendCvmPin(pinBlock, true);

cancelPin

Item	Details
Signature	void cancelPin()
Inputs	None
Outputs	None
Callback	onRequestSetPin

Item	Details
Usage Example	pos.cancelPin();

pinMapSync

Item	Details
Signature	void pinMapSync(String datas, int timeout)
Inputs	**datas**: the string of keyboard pin coordinate position **timeout**: timout of pin input
Outputs	None
Callback	onQposRequestPinResult
Usage Example	pos.pinMapSync(value,30);

sendOnlineProcessResult

Item	Details
Signature	void sendOnlineProcessResult(String tlv)
Inputs	**tlv**: online processing result
Outputs	None
Callback	onRequestOnlineProcess
Usage Example	pos.sendOnlineProcessResult("8A023030");

getNFCBatchData

Item	Details
Signature	Hashtable<String, String> getNFCBatchData()
Inputs	None
Outputs	Return the TLV data of NFC
Callback	None

Item	Details
Usage Example	Hashtable<String, String> data = pos.getNFCBatchData();

4.4.3 EMV Configuration APIs

The following describes the emv config update interface in detail.

API Methods	Description
updateEMVConfigByXml(String xmlContent)	Update the emv config

updateEMVConfigByXml

Item	Details
Signature	void updateEMVConfigByXml(String xmlContent)
Inputs	**xmlContent**: parse EMV configuration file in xml format to get string
Outputs	None
Callback	onReturnCustomConfigResult
Usage Example	pos.updateEMVConfigByXml(new String(FileUtils.readAssetsLine("emv_profile_tlv.xml”)));

4.4.4 Key Management APIs

These APIs are used to update encryption keys (e.g., Master Key, Work Key, IPEK) on the QPOS device, ensuring transaction security.

API Methods	Description
updateKeyByTR_31(int keyIndex, String keyBlock)	Update the device key according analysis the TR31 keyblock

updateKeyByTR_31

Item	Details
Signature	updateKeyByTR_31(int keyIndex, String keyBlock)

Item	Details
Inputs	**keyIndex**: new key index, from 0 start **keyBlock**: TR31 keybolck
Outputs	None
Callback	onReturnUpdateKeyByTR_31Result
Example	pos.updateKeyByTR_31(0,"B0080B1TX00E00007ADC3F4EC9DEC96726CAB25032AC86B71581429795F43E2B6B5DBCD646FE8F3A");

4.4.5 Firmware Update APIs

These APIs are used to update the firmware of the QPOS device, ensuring compatibility with new features and security patches.

updatePosFirmware

Item	Details
Signature	int updatePosFirmware(byte[] data, String deviceAddress)
Inputs	**data**: the firmware asc file datas. **deviceAddress**: deevice address, like bluetooth connection, need input the device bluetooth mac address, other connection method, can input “”.
Outputs	Return int value,
Description	Update the device firmware: -1 means error, 0 means can update.
Callback	onUpdatePosFirmwareResult

getUpdateProgress

Item	Details
Signature	int getUpdateProgress()
Inputs	None
Outputs	Return the firmware update progress integer value
Description	Get the device firmware update progress. And it should be called in the child thread so that avoid to block the main thread.

4.4.6 Device Information APIs

These APIs are used to retrieve hardware and software information from the QPOS device.

getQposId

Item	Details
Signature	void getQposId()
Inputs	None
Outputs	None
Description	Retrieve the POS_ID of the QPOS device. Results are returned by onQposIdResult.
Callback	onQposIdResult

getQposInfo

Item	Details
Signature	void getQposInfo()
Inputs	None
Outputs	None
Description	Get the device info
Callback	onQposInfoResult

4.5 Delegate Methods Reference

Delegate methods (callbacks) are triggered by the SDK to notify the application of event status(e.g., connection results, key update results). All callbacks are implemented via the `QPOSServiceListener` interface.

4.5.1 Connection Callbacks

These callbacks notify the application of device connection/disconnection status.

onRequestQposConnected

Item	Details
Signature	void onRequestQposConnected ()
Inputs	None
Outputs	None
Description	Callback of connected success

onRequestQposDisconnected

Item	Details
Signature	void onRequestQposDisconnected()
Inputs	None
Outputs	None
Description	Callback of disconnect

onRequestNoQposDetected

Item	Details
Signature	void onRequestNoQposDetected()
Inputs	None
Outputs	None
Description	Callback of connected fail

4.5.2 Transaction Callbacks

onRequestSetAmount

Item	Details
Signature	void onRequestSetAmount()
Inputs	None

Item	Details
Outputs	None
Description	Callback of setAmount

onRequestTime

Item	Details
Signature	void onRequestTime()
Inputs	None
Outputs	None
Description	Callback method of request user to set time

onRequestSelectEmvApp

Item	Details
Signature	void onRequestSelectEmvApp(ArrayList<String> appList)
Inputs	**appList**: the emv applications list
Outputs	None
Description	Callback method of request user to select emv app

onDoTradeResult

Item	Details
Signature	void onDoTradeResult(DoTradeResult result,Hashtable<String, String> decodeData)
Inputs	**result**: DoTradeResult.NONE, MCR, ICC, NOT_ICC, BAD_SWIPE, NO_RESPONSE, NO_UPDATE_WORK_KEY, NFC_ONLINE, NFC_OFFLINE, NFC_DECLINED, TRY_ANOTHER_INTERFACE, CARD_NOT_SUPPORT, PLS_SEE_PHONE, REQUEST_ONLINE **decodeData**: the transaction data
Outputs	None
Description	Callback of track data response

onRequestSetPin

Item	Details
Signature	void onRequestSetPin()
Inputs	None
Outputs	None
Description	Callback of sendPin()/sendCvmPin(), this is used for CR100

onRequestSetPin

Item	Details
Signature	void onRequestSetPin(boolean isOfflinePin, int tryNum)
Inputs	**isOfflinePin**: true or false to describe whether PIN is offline PIN **tryNum**: remaining input times for offline pin
Outputs	None
Description	This is used to prompt user entry PIN on pinpad of D70.

onQposRequestPinResult

Item	Details
Signature	void onQposRequestPinResult(List<String> dataList, int offlineTime)
Inputs	**dataList**: the random number value list returned by POS **offlineTime**: remaining input times for offline pin
Outputs	None
Description	This is used for smart POS to let app draw the keyboard and send key position to POS. The callback of pinMapSync().

onReturnGetPinInputResult

Item	Details
Signature	void onReturnGetPinInputResult(int num)
Inputs	**num**: the num is the counter of your pin input
Outputs	None

Item	Details
Description	This is used for smart POS to return the number of pin inputs

onRequestOnlineProcess

Item	Details
Signature	void onRequestOnlineProcess(String tlv)
Inputs	**tlv**: contains the tag-length-value data structure returned by the EMV kernel
Outputs	None
Description	EMV kernel request the online handler, should send the results back to EMV kernel by sendOnlineProcessResult()

onRequestTransactionResult

Item	Details
Signature	void onRequestTransactionResult(TransactionResult transactionResult)
Inputs	**transactionResult**: TransactionResult.APPROVED, TERMINATED, DECLINED, CANCEL, CAPK_FAIL, NOT_ICC, SELECT_APP_FAIL, DEVICE_ERROR, CARD_NOT_SUPPORTED, MISSING_MANDATORY_DATA, CARD_BLOCKED_OR_NO_EMV_APPS, INVALID_ICC_DATA, FALLBACK, NFC_TERMINATED, CARD_REMOVED, TRADE_LOG_FULL, TRANSACTION_NOT_ALLOWED_AMOUNT_EXCEED, CONTACTLESS_TRANSACTION_NOT_ALLOW, TRANS_TOKEN_INVALID, CARD_BLOCKED, APP_BLOCKED, MULTIPLE_CARDS
Outputs	None
Description	Transaction response

onRequestBatchData

Item	Details
Signature	void onRequestBatchData(String tlv)
Inputs	**tlv**: contains the tag-length-value data structure returned by the EMV kernel
Outputs	None
Description	Response of ICC transactions: when the transaction is finished. The batch data will be returned to the application

onReturnReversalData

Item	Details
Signature	void onReturnReversalData(String tlv)
Inputs	**tlv**: contains the tag-length-value data structure returned by the EMV kernel
Outputs	None
Description	Ic card reversal data response

onEmvICCExceptionData

Item	Details
Signature	void onEmvICCExceptionData(String tlv)
Inputs	**tlv**: contains the tag-length-value data structure returned by the EMV kernel
Outputs	None
Description	Emv ICC Exception response

4.5.3 EMV Configuration Callbacks

onReturnCustomConfigResult

Item	Details
Signature	void onReturnCustomConfigResult(boolean isSuccess, String result)
Inputs	**isSuccess**: result of the emv update **result**: if update failed, it will return the specific error reasons
Outputs	None
Description	Callback of updateEMVConfigByXml()

4.5.4 Key Management Callbacks

These callbacks notify the application of key update status.

onReturnUpdateKeyByTR_31Result

Item	Details
Signature	void onReturnUpdateKeyByTR_31Result(boolean result)
Inputs	**result**: true– update key success, false – update key failure
Outputs	None
Description	Callback of updateKeyByTR_31()

4.5.5 Firmware Update Callbacks

onUpdatePosFirmwareResult

Item	Details
Signature	void onUpdatePosFirmwareResult(UpdateInformationResult result)
Inputs	**result**: update success, return UPDATE_SUCCESS； Update failed, return UPDATE_FAIL； Update firmware package is worng, return UPDATE_PACKET_VEFIRY_ERROR
Outputs	None
Description	Callback of updatePosFirmware()

4.5.6 Device Information Callbacks

These callbacks return the device information retrieved by the SDK.

onQposIdResult

Item	Details
Signature	void onQposIdResult (Hashtable<String, String> posIdTable)
Inputs	**posIdTable**: posId - device sn psamId,merchantId,vendorCode
Outputs	None
Description	Callback of getQposId()

onQposInfoResult

Item	Details
Signature	void onQposInfoResult (Hashtable<String, String> posInfoTable)
Inputs	**posInfoTable**: isSupportedTrack1,isSupportedTrack2,isSupportedTrack3, bootloaderVersion,firmwareVersion,isUsbConnected,batteryLevel,hardwareVersion, PCI_firmwareVersion,PCI_hardwareVersion,compileTime
Outputs	None
Description	Callback of updatePosFirmware()