



# **Payment Device SDK for Android Axium Smart POS Supplement**

For Software Version 5.3.1 (Andromeda)

Document Version: 2.0.6

Date: 19th June 2026

# TABLE OF CONTENTS

TABLE OF CONTENTS	1
1. Introduction	4
2. High level Overview	5
3. Preparing for Development	6
Pre-requisites	6
Android Flavours	7
Lite	7
Full	7
Device Setup	8
Installing Apps	8
Wi-Fi Configuration	8
Bluetooth Configuration	9
Bluetooth Device Name	9
Supported Devices	10
Axiom RX5000	10
Setting Up the Device	10
Restarting the Device	10
Axiom EX4000, Axiom EX6000 and Axiom EX8000	10
Setting Up the Device	10
Charging the Device	10
Restarting the Device	10
Axiom DX4000 and DX8000	11
Setting Up the Device	11
Restarting the Device	11
Charging the DX4000 Device	11
Charging the DX8000 Device	11
4. Smart POS API Updates	12
Initialize	12
Parameters Required for a Smart POS Initialize	12
Set Properties	12
Start Transaction	13
Parameters Required for a Smart POS Transaction	13
Parameters That Will Cause Errors for a Smart POS Transaction	13
New Parameters Available for a Smart POS Transaction	14
ParameterKeys.OfflineTransactionUploadMode	14
ParameterKeys.MagstripeDebit	14
ParameterKeys.MagstripeSignature	14
ParameterKeys.PartialApproval	15
ParameterKeys.TipAmount	15
Events	16
Transaction Update Events	16

TransactionUpdate	16
New Error Codes	16
Start Transaction	16
Transaction Finished	16
Offline Transaction Queue Processing	17
Device Disconnection During A Transaction	17
Retrieving Log Files	19
<b>5. OTA Application Updates</b>	<b>20</b>
Triggering update via Axiom Device	20
Triggering update via ChipDNA Mobile	20
Appendix 1 - Supported API Methods	22
Appendix 2 - Smart POS Unsupported Features	24

## Document History

Document Version	Software Version	Date yyyy-mm-dd	Summary of Changes
1.0.0	4.1.0	2025-09-17	Document Created
1.0.1	4.1.0	2025-10-02	Added information on how to change the bluetooth name of an Axiom device. Add new error code.
2.0.0	5.0.0	2025-11-14	Updated version of live pilot. Added section on how to retrieve log files. Updated supported features. Added information on how disconnections are handled during a transaction. Updated information around device bluetooth name.
2.0.1	5.0.1	2026-01-09	Updated for PAN Key entry support
2.0.2	5.0.1	2026-01-28	Added warning that remote linked refunds are not supported on Axiom devices.
2.0.3	5.1.0	2026-02-11	Support for linked refunds added on smart POS integrations.

Refunds on smart POS integrations don't support auto confirm, they are always confirmed and so transaction status will indicate if a transaction needs to be confirmed.

2.0.4	5.2.0	2026-03-25	Update document version. Update to high level overview. Added section for OTA updates.
2.0.5	5.3.0	2026-05-01	Updated documentation version.
2.0.6	5.3.1	2026-06-19	Updated documentation version. Removed live pilot restriction around deferred auth preference flag.

# 1. Introduction

This supplement document contains additional information about how to set up Smart POS (Point Of Sale) functionality in the Payment Device SDK.

This document details any pre-requisites required, configuring your development environment, and how to prepare your application for release. It also goes through the Smart POS integration specific API changes that integrations will need to be aware of.



If you're intending to develop an application that runs on the Axiom device you should use this document in conjunction with the Axiom Development Guide Supplement

## 2. High level Overview

The component diagram shows the interactions between the components in a Smart POS integration with the Payment Device SDK.

The mobile device hosting the integrating application will require a bluetooth connection to the Ingenico Axiom device, to communicate with the NMI Android All In One (AAIO) service.

When integrating in All in one mode connection to the NMI AAIO service operations of local TCP/IP connection.

The AAIO service communicates with the Axiom Payment Service to run transactions on the Ingenico Axiom Device and requires an internet connection to the NMI Backend to process the payments.

The Axiom Payment service requires an internet connection to the Ingenico Hardware Estate Management (HEM), to monitor and update software on the Ingenico Axiom device.

This differs from existing Payment Device SDK integrations, where the mobile device required the internet connection to process payments with the NMI Backend.

*Diagram 1 – Components of Smart POS Local Semi-Integration*

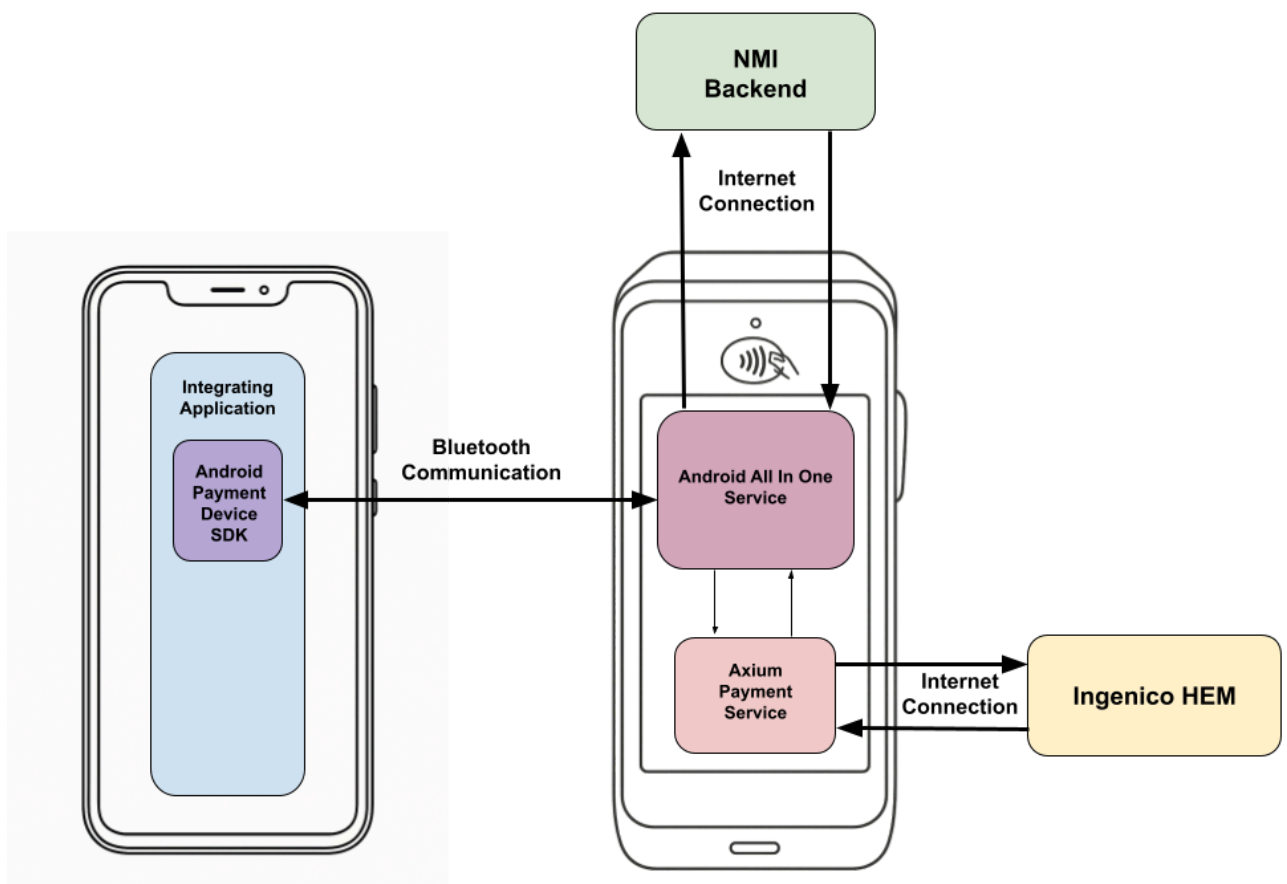
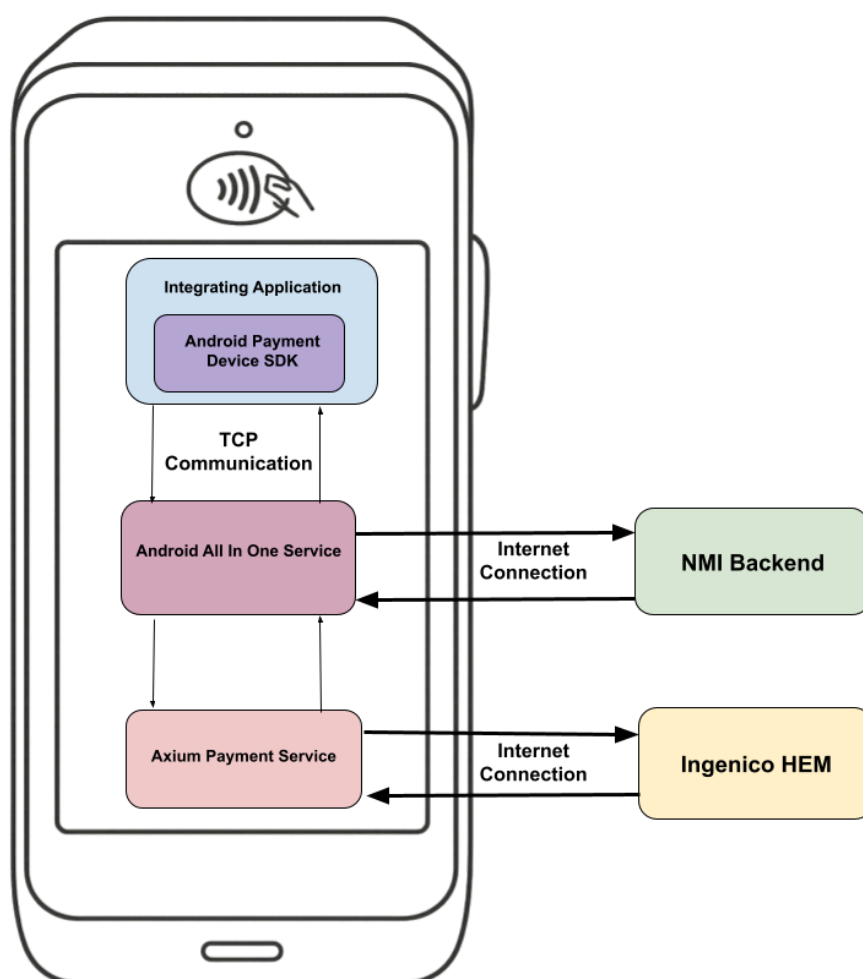


Diagram 2 – Components of Smart POS All In One Integration



### 3. Preparing for Development



If intending to integrate with the Axium device in All in One mode please also refer to the Axium Development Guide Supplement

#### Pre-requisites

The SDK software is packaged and supplied as .zip archives, where the contents are described in [Table 1](#).

*Table 1 – Contents of NMI Payment Device for Android .zip archive.*

Folder Name	Description
ChipDnaMobileJavaDemo	An example client written in Java, including source code, demonstrating a simple integration using the ChipDNA Mobile SDK.
ChipDnaMobileKotlinDemo	An example client written in Kotlin, including source code, demonstrating a simple integration using the ChipDNA Mobile SDK.
doc	The documentation of the ChipDNA Mobile API.
libs	The ChipDNA Mobile SDK binaries.

## ANDROID FLAVOURS

Within the `libs` folder, there are two further folders, representing the two flavours of the ChipDNA Mobile SDK;

- `libs/lite`
- `libs/full`



**Do not include both SDK flavours in the same project. Choose the one that best fits your integration approach. When enabling Smart POS Integration, all other POS devices will be unavailable.**

### Lite

The Lite flavour has a minimum OS of Android 10 (SDK version 29) and does not support the use of Tap To Pay.

The ChipDnaMobile jar file has a different name of `ChipDnaMobile-lite.jar`.



**The Lite Flavour is recommended for Smart POS Integrations.**

### Full

The Full flavour has a minimum OS of Android 12 (SDK version 31) and does support the use of Tap To Pay, requiring the extra dependencies of the Cloud Commerce SDK.



# Device Setup

## INSTALLING APPS

Your Axiom device will come with all the applications and firmware needed to accept payments already installed.

The following apps should be installed on your device when you receive it, there may be some additional apps as well.

- AppStore
- AXIUM Retail Core
- ARC Comm Config
- AXIUM UX
- View Security Settings App
- TransactionInitiatorUI

If any of these are not present please contact integration support.



**Ingenico Axiom devices run on a security-hardened version of Android, so while they operate in a similar way to standard Android devices, additional security measures are enforced.**

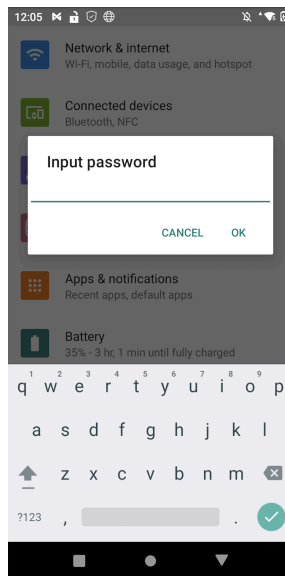
## WI-FI CONFIGURATION



**Ingenico Axiom devices must have an internet connection for transactions to be processed.**

To connect the device to a Wi-Fi network, either pull down on the top task bar and select the Wi-Fi symbol, or open the Settings application and select Network and Internet.

When a password is required, enter **350000**. Once the password is entered, the device can be configured to connect to a Wi-Fi network.



## BLUETOOTH CONFIGURATION

To connect to the device via bluetooth:

1. Turn the payment device on, and enable bluetooth.
  - a. Navigate to the settings app.
  - b. Select Connected devices.
  - c. Select Connection preferences.
  - d. Select Bluetooth.
  - e. Check the top On/Off bar below the title:
    - i. If it is grayed out and displaying Off, tap the switch once to turn Bluetooth on.
    - ii. If it is already highlighted and displaying On, Bluetooth is enabled.
2. On the mobile device, within the mobile device's settings menu, navigate to Connected Devices → pair new device and select the payment device. Both the mobile device and payment device should display a pairing prompt, with matching bluetooth pairing codes. Select pair on each device.
3. The payment device will now appear in the list of available payment devices within the SDK when the method `getAvailablePinPads()` is called.

## Bluetooth Device Name

When the Transaction Initiator application is first installed on an Axiom device, it will set the bluetooth name of the device to the following format:

**AXIUM-{model number}-{serial number}**, e.g. **AXIUM-DX6000-235GKD420151**.

# Supported Devices

The following Axiom Smart POS devices are supported:

## AXIUM RX5000

### Setting Up the Device

1. **Connect the HDMI cable** to the port on the back of the device.
2. **Use the split cable:**
  - Plug the **power pin** into a power socket.
  - *(The USB connector is not required for normal use, as connections are made via bluetooth.)*

### Restarting the Device

1. Press and hold the **Green “OK” button** and the **dot ( . ) button** at the same time.
2. Two tiles will appear at the top of the screen:
  - **Services** (not normally needed)
  - **Restart** (select this option)

## AXIUM EX4000, AXIUM EX6000 AND AXIUM EX8000

### Setting Up the Device

1. Press the **orange power button** on the right side of the device to turn it on.

### Charging the Device

1. A **USB-C port** is located on the left side of the device for charging.
2. Connect the charging cable to this port until the device is fully charged.

### Restarting the Device

1. Press and hold the **power button** for a few seconds.
2. A menu will appear with the following options:
  - **Power off**
  - **Restart** (select this option)

- **Screenshot**
- **Services** (not normally needed)



Ingenico Axium EX4000 devices cannot perform magstripe transactions.

## AXIUM DX4000 AND DX8000

### Setting Up the Device

1. Press the **power button** on the left side of the device to turn it on.

### Restarting the Device

1. Press and hold the **power button** for a few seconds.
2. A menu will appear with the following options:
  - a. **Power off**
  - b. **Restart** (select this option)
  - c. **Screenshot**
  - d. **Services** (not normally needed)

### Charging the **DX4000** Device

1. **Remove the back cover:**
  - Push the tab located on the back of the device upwards.
  - Lift off the panel to reveal the battery and USB-C charging port.
2. **Connect the charging cable:**
  - Under the battery, plug the cable into the USB-C port.
3. **Replace the back cover:**
  - The device has a **groove on the left side** for the charging cable to fit through.
  - Snap the cover back into place with the cable running through the groove.

### Charging the **DX8000** Device

3. A **USB-C port** is located on the left side of the device for charging.

4. Connect the charging cable to this port until the device is fully charged.

## 4. Smart POS API Updates

### Initialize

Your application interacts with the NMI Payment Device SDK using the method in the `ChipDnaMobile` object instance. Before beginning the messaging process, you must initialize a new `ChipDnaMobile` instance.

This section describes the parameters required for initialising the SDK for use with Smart POS, the parameters that will cause errors if used.

This method returns a `Parameters` object with the result of the initialization. Any errors encountered will also be reported in this object.

```
Parameters initialize(Context, Parameters)
```

### PARAMETERS REQUIRED FOR A SMART POS INITIALIZE

The following parameters are required to initialize with Smart POS integration:

- `ParameterKeys.Password` - Database encryption password
- `ParameterKeys.IsSmartPosIntegration` -
  - `TRUE` enables SmartPos integration, allowing use of the Axiom device only.
  - `FALSE` or omitting parameter, enables the standard device integration, which allows the use of all other supported PinPads.



**GetAvailablePinPads** returns only the PIN pads supported by the currently enabled integration. If `IsSmartPosIntegration` is enabled, then only Axiom devices will be returned in the response.

### Set Properties

There are no additional requirements around the `setProperties` method required to Smart POS transactions beyond the usual:

- `TerminalId`
- `TerminalKey`
- `External Device Info` (retrievable from `getAvailablePinPads`)
  - `ParameterKeys.PinPadName`
  - `ParameterKeys.PinPadConnectionType`
- `Environment`

The following existing Parameter keys will be accepted, but will have no affect when using a Smart POs integration:

- `ApplicationIdentifier`

## Start Transaction

This section describes the parameters required for a Smart POS transaction, the parameters that will cause errors if used.

### PARAMETERS REQUIRED FOR A SMART POS TRANSACTION

The following parameters are required to start a Smart POS transaction:

- `ParameterKeys.Amount`
- `ParameterKeys.UserReference`
- `ParameterKeys.PaymentMethod`
  - **Must be set to** `ParameterValues.Card`
- `ParameterKeys.TransactionType`
- `ParameterKeys.Currency`
  - **Retrievable from** `getAvailableCurrencies`

### PARAMETERS THAT WILL CAUSE ERRORS FOR A SMART POS TRANSACTION

Smart POS currently only supports a limited number of features compared to traditional POS devices, therefore the following parameters will return an error if included when starting a Smart POS transaction:

- `ParameterKeys.DelayOnlineProcessing`
  - `ParameterValues.TRUE`
- `ParameterKeys.TippingType`
  - `ParameterValues.OnDeviceTipping`
  - `ParameterValues.BothTipping`
- `ParameterKeys.PurchaseOrderNumber`
- `ParameterKeys.TaxAmount`
- `ParameterKeys.TransactionPOI`
  - `ParameterValues.TapToMobile`
- `ParameterKeys.OnDeviceTippingPrompt`
- `ParameterKeys.DynamicTippingAmounts`
- `ParameterKeys.DynamicTippingPercentages`
- `ParameterKeys.DynamicTippingHeader`
- `ParameterKeys.FeatureTokens`
- `ParameterKeys.CredentailOnFileFirstStore`
- `ParameterKeys.CredentailOnFileReason`

## NEW PARAMETERS AVAILABLE FOR A SMART POS TRANSACTION

The following parameters are now available when starting a Smart POS transaction. If they are not passed, then the default values specified below are used.

`ParameterKeys.TransactionAuthPreference`

Determines how the transaction authorisation is initially processed for Smart POS transactions and can have the following values, (Default is `Online`) :

- `ParameterValues.Online`
  - Attempts online authorization, declines if online fails.
- `ParameterValues.DeferredForced`
  - Immediately processes transaction offline, skipping online authorization.
- `ParameterValues.OnlinePreferred`

Attempts online authorization, processes transaction offline if online fails.

### `ParameterKeys.OfflineTransactionUploadMode`

Determines how offline transactions are uploaded for processing for Smart POS transactions and can have the following values, (Default is `Automatic`) :

- `ParameterValues.Automatic`
  - Background worker automatically processes offline transactions when network returns
- `ParameterValues.Manual`
  - Requires a call to `processOfflineRequest` providing the `ParameterKeys.UserReference` for the transaction to attempt to be uploaded to the backend.

### `ParameterKeys.MagstripeDebit`

Determines if Debit is enabled for Magstripe Smart POS transactions and can have the following values, (Default is `True`) :

- `ParameterValues.True`
- `ParameterValues.False`

### `ParameterKeys.MagstripeSignature`

Determines if signature is enabled for Magstripe Smart POS transactions and can have the following values, (Default is `True`) :

- `ParameterValues.True`
- `ParameterValues.False`

## ParameterKeys.PartialApproval

Determines if partial approval is enabled for Smart POS transactions and can have the following values, (Default is `False`) :

- `ParameterValues.True`
- `ParameterValues.False`

## ParameterKeys.TipAmount

As On Device tipping is not supported when using a Smart POS device, a tip amount can be passed into the start transaction request. This parameter key will accept the value in the same format as `ParameterKeys.Amount`.

## Methods Requiring Smart POS Device Connection

The following API methods will require a connection to the Smart POS Payment device in order to succeed, unlike when used in a standard device integration:

- `ConfirmTransaction`
- `VoidTransaction`
- `LinkRefundTransaction`
- `GetTransactionInformation`
- `GetStatus` (For a full set of parameter keys to be returned)



# Events

## Transaction Update Events

This section describes the events returned by the SDK during operation of Smart POS, including those for error codes and transaction events.

### TRANSACTIONUPDATE

The following new event is emitted during the `transaction` process:

`AttemptingToReconnect` indicates that the Payment Device SDK has lost connection to the Smart POS device and is attempting to reconnect to retrieve the transaction result.

## New Error Codes

This section details the new error codes added as part of the new Smart POS integration. A full list of error codes can be viewed within the generated web documentation supplied within the release bundles docs directory. Navigate to `index.html` then under Modules > `ChipDnaMobileErrorCodes`.

### START TRANSACTION

The following new error codes may be returned during calls to `startTransaction`.

- `TransactionAuthPreferenceInvalid`: Indicates that the value passed with `ParameterKeys.TransactionAuthPreference` is invalid.
- `OfflineTransactionUploadModeInvalid`: Indicates that the value passed with `ParameterKeys.OfflineTransactionUploadMode` is invalid.

### TRANSACTION FINISHED

The following new error codes may be returned in the `transactionFinishedListener`.

- `TransactionInformationUnavailable` Indicates the transaction information could not be retrieved due to a connection failure with the Smart POS device.



The `AUTO_CONFIRM` parameter is not supported for Standalone Refunds on Axiom devices.

You should refer to `TRANSACTION_RESULT` and `TRANSACTION_STATUS` values returned in the Transaction Finished and Transaction Information events to determine whether a `ConfirmTransaction` call is expected to succeed.

Transactions with a `TRANSACTION_STATUS` of `Committed` are already confirmed and will not succeed if `ConfirmTransaction` is called.

`ConfirmTransaction` calls on a transaction with a `TRANSACTION_RESULT` of `Declined` will also fail.

# Offline Transaction Queue Processing

Only transactions associated with the currently configured environment can be processed in the offline queue. Therefore if configured on Staging, only offline transactions performed against Staging credentials will be able to be processed. Likewise if configured on Live, only offline transactions performed against Live credentials will be able to be processed

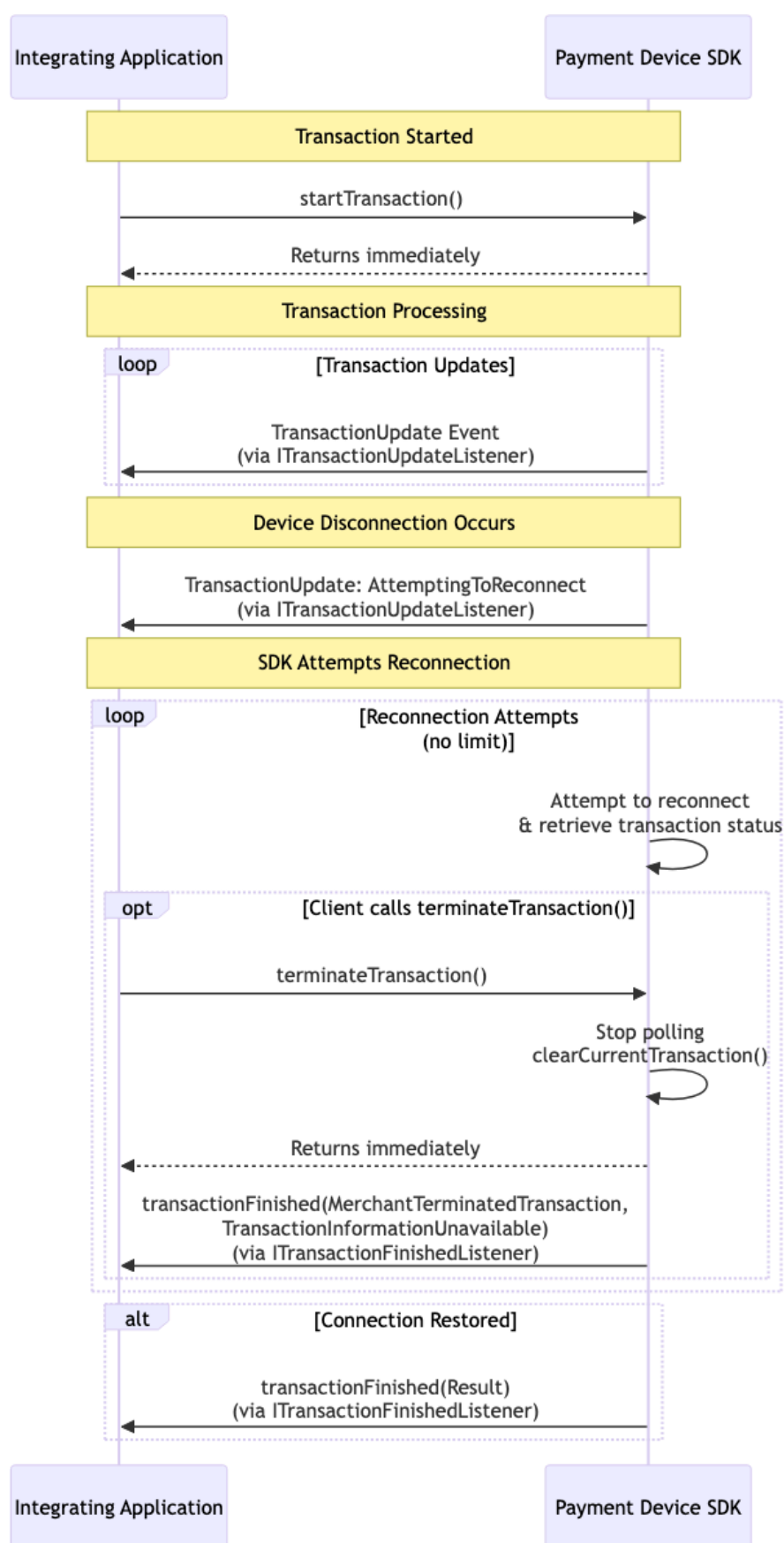
The environment associated with a transaction can be found by calling `getTransactionInformation` for the specified transaction looking at the value associated with `ParameterKeys.Environment`. The currently configured environment can be found by calling `getStatus` where the currently configured environment is found by looking at the value associated with `ParameterKeys.Environment`.

## Device Disconnection During A Transaction

As described in the [high level overview](#), transaction processing is fully performed on the Ingenico Axium device. Therefore if a disconnection between the Payment Device SDK and device occurs during a transaction, a termination does not automatically occur, as it would have done with a non Smart POS integration.

If a disconnection between the Payment Device SDK and device occurs during a transaction, then the Payment Device SDK will attempt to reconnect to the Axium device, until a reconnection is made or the `terminateTransaction` API method is called by the integrating application, as shown in the sequence diagram below.

Diagram 2 – Sequence diagram of a device disconnection during a transaction



If the `terminateTransaction` API method is called by the integrating application during the disconnection, the `transactionFinished` event will be returned including the following parameters:

- `TransactionResult` = `Unknown`
- `TransactionState` = `Unknown`
- `Errors` = `MerchantTerminatedTransaction`, `TransactionInformationUnavailable`

When this occurs, the integration application will need to call `getTransactionInformation` once connection to the Ingenico Axiom device has been restored to see the final outcome of the transaction.

## Retrieving Log Files

When integrated into a Smart POS device, local log files can be retrieved to aid integration support in diagnosing issues.

The below code snippet gives the file location of logs for Smart POS integrations.

```
String chipDNASmartPosLogPathComponent =  
"NMI"+File.separator+"ChipDNA"+File.separator+"smartpos_logs"+File.separator
```

## 5. OTA Application Updates

NMI has implemented over-the-air (OTA) application updates. As part of this, a new NMI Device Agent application will be installed on your device. This has been designed to align with the existing Miura OTA update experience while leveraging the additional capabilities of Smart POS devices.

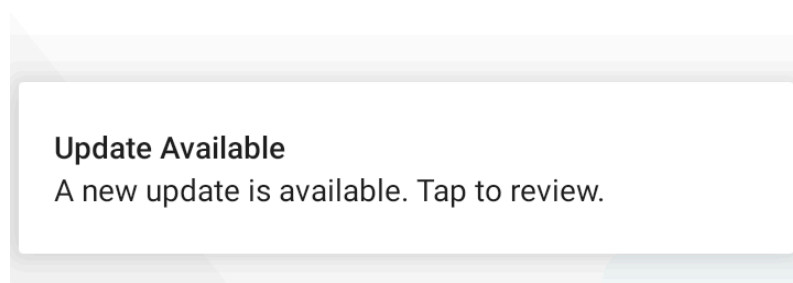
When an OTA update becomes available, your application will receive a Firmware Update event. A notification will also be displayed on the device. You can either trigger the update programmatically via the SDK or initiate it directly on the Axiom device by tapping the notification.

### Triggering update via Axiom Device

As with the ChipDNA Mobile SDK update process an update can either be of the type:

- Required - The update must be performed before transaction processing can continue.
- Deferrable - The updates can be deferred for later but will expire eventually to become required.

There are 2 ways to trigger the update via the Axiom device UI. You can either tap the notification which will open the device agent application, or you can manually enter the device agent application and trigger the update from there.



### Triggering update via ChipDNA Mobile

When triggering the update using the ChipDNA Mobile SDK the NMI Device Agent application will take over and will immediately begin updating. Once the update has completed the previous app will automatically come to the foreground.

The ChipDNA Mobile API remains unchanged from existing Miura integrations. For implementation details, refer to the API Developer Integration Guide under:

- Section 6: *Configuration and Utility Methods* (specifically *Connect and Configure*)
- Section 7: *FirmwareUpdate Callback*

## Update Available

Required by: 29 Mar 2026 at 11:58

Update Now

Dismiss

*Device Agent - Update Available*

## Update Required!

Install the required update before taking payments.

Update Now

*Device Agent - Update Required*



Large updates can take upwards of 5 to 10 minutes to complete. Unfortunately the Download and Install process are controlled by the Axium firmware which doesn't return progress updates. If an update fails a screen will be displayed allowing you to Retry the update.

# Appendix 1 - Supported API Methods

This table shows all API methods in the Payment Device SDK and shows their supported status with Smart POS integrations.

*Table 2 - Supported Smart POS API Methods*

API Method	Supported	Notes
initialize	Yes	Requires passing of <code>IsSmartPosIntegration</code> parameter key with value of <code>TRUE</code>
setProperties	Yes	-
connectAndConfigure	Yes	-
getAvailableCurrencies	Yes	Supports GBP and USD
getAvailablePinPads	Yes	-
getStatus	Yes	Requires connection to payment device for full status
getTransactionInformation	Yes	Requires connection to payment device
getMerchantData	No	-
requestTmsUpdate	Yes	-
disconnect	Yes	-
dispose	Yes	-
startTransaction	Yes	-
continueDeferredAuthorization	No	-
continueForcedAcceptance	No	-
continuePartialApproval	Yes	-
continueSignatureVerification	Yes	-
continueVoiceReferral	No	-

continueSignatureCapture	No	Signature capture performed on payment device
continueApplicationSelection	No	-
confirmTransaction	Yes	Requires connection to payment device
voidTransaction	Yes	Requires connection to payment device
linkedRefundTransaction	Yes	Requires connection to payment device
getCardDetails	Yes	-
terminateTransaction	Yes	-
processReceipt	No	-
setIdleMessage	No	-
deleteFailedOfflineRequest	No	-
retryFailedOfflineRequest	No	-
processOfflineRequest	Yes	-



# Appendix 2 - Smart POS Unsupported Features

This table details the transaction features that are not supported on Smart POS integrations compared to Non-Smart POS integrations.

Table 3 - Supported Smart POS Features

Feature	Supported
Delay Online Processing	No
On Device Tipping	No
One Way Card Tokens	No