ROAMpayX Web App Integration Guide

Android ONLY

Version 2.3
April 2021

Revision History

Version Number	Revision Date	Revision Description
v1.0	8//2019	Initial release.
v1.1	9/2019	Added targeted_platform in the Security Token API Request.
v2.0	1/2020	Includes detailed descriptions of all required API calls, updated workflow diagram & direct link to mPOS SDK documentation. Minor revisions included throughout.
v2.1	2/2020	Added details for Device Serial Number based login.
v2.2	2/2020	Added example for Partial Auth, Decline & Reversal.
v2.2.1	2/2020	Updated Code Snippet in Section 3_4
v2.2.2	3/2020	Added example for Retrieving a Transaction Response JSON via Intent Added Example for Response to Pre-Login errors Updated Invalid or Expired Session Return Code to 4995
v2.2.3	4/2020	Added introductory section 1_2, as well as updated parameter information for starting a payment and code snippets in Section 3_4.
v2.3	4/2021	Added support for tenderType to allow manually-entered and cash transactions.

Table of Contents

1 Introduction

The objective of this document is to provide the necessary steps for getting started with payment acceptance using ROAMpay X (RPX). Deeplink functionality allows native Android or web applications to utilize RPX for payment processing using supported Ingenico devices.

1_1 What is Deeplink?

RPX Deeplink is a secured, simplified and seamless method for card acceptance with authenticated third-party applications. Deeplink provides a semi-integrated mode through the RPX mobile application to allow an authenticated third-party app to integrate with RPX through use of Inter-App Deeplinking methodology.

UseCase : ISVs with their own mobile POS app can use RPX Deeplink for an easier, simpler and quicker option to add payment acceptance functionality from within their own app – minus the overheads of development, testing and subsequent publishing for deployment of new versions.

UseCase : ISVs with only a website/portal will be able to leverage RPX Deeplink to accept Card-Present transactions without requiring the development of a mobile application.

Deeplink:

- Eliminates implementing support for the underlying mPOS SDK for Native mobile apps.
- Drives Card-Present interactions from an authenticated Web App using a mobile browser – no mobile app needed.
- Manages device (reader, printer, etc.) connectivity and reader firmware upgrades.
- Provides speedier and easier implementation of payment acceptance.
- Supports an iterative approach to add feature when required.

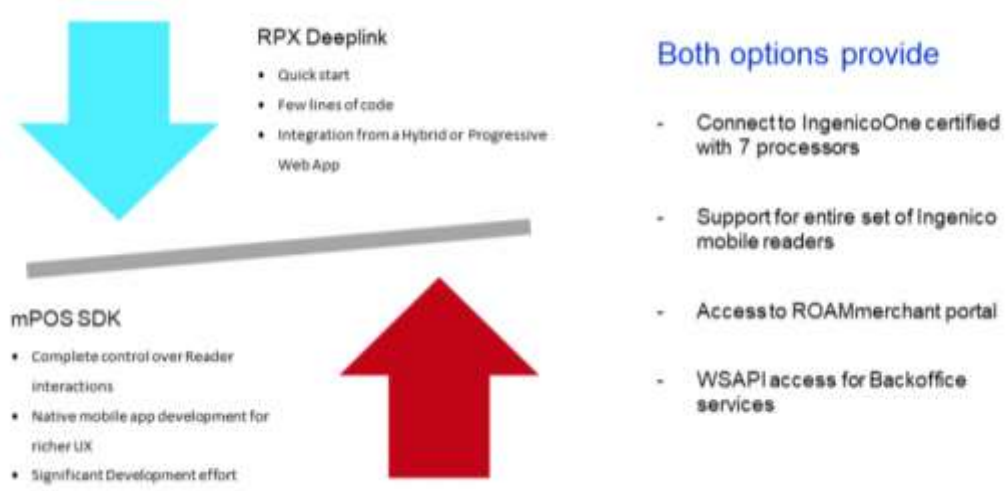

Figure 1: Deeplink vs mPOS SDK Integration

1_2 Requirements

- Android device must be running Android 5.0 (API 21) or higher
- Android device must have Google Play services v16.0.0 or higher installed
- Android device must have Locked bootloader
- Android device must be unrooted
- Android Device must have a stock ROM

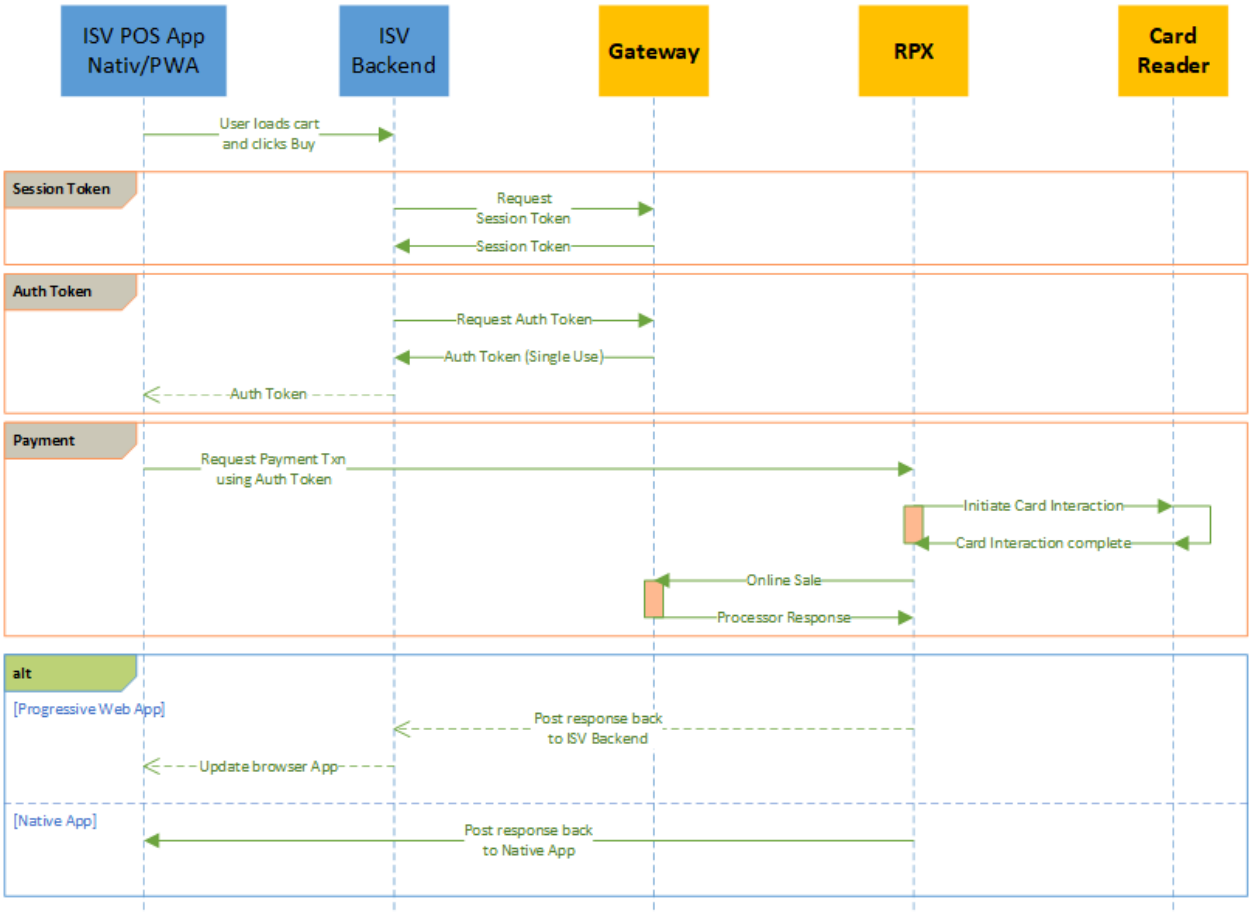

Figure 2: Deeplink Transaction Workflow

2 Integration

1. Before you begin integrating with RPX for deep link support, you must first contact Ingenico API Support to set up:
    a. a user account,
    b. application token,
    c. preferred login strategy used by ROAMpay X, and
    d. register callback URL and shared secret (used for posting transaction response for the payment processed through ROAMpay X).

    This can be done on our Ingenico Service Desk portal available here. Once you have registered a callback URL and obtained an application token, you can proceed with integration with next steps.

2. Acquire the APK files from Ingenico (these files must be downloaded directly from mobile device).

3 Transaction

3_1 Session Token

Using the Ingenico Authorization REST API, log into the system with the username and password to retrieve a session (Authorization) token. In order to begin utilizing Ingenico services, you must first log in and retrieve a session token. This is performed by passing the request as shown below.

Headers		
Parameter	**Required**	**Description**
`X-Roam-Key`	Yes	Application token that identifies the request origin. This key is unique to every client, as well as every environment. If you do not have an application token, please contact API Support here.
`X-Roam-ApiVersion`	Yes	Version of the API being used (e.g., 2.0.0).
`X-Roam-ClientVersion`	Yes	Client version (can vary by application). This is set by developers (e.g., 1.0.0).
`Content-Type`	Yes	Content type (e.g., application/json).

Headers

```
Request URL:

 https://mcm.roamdata.com/wsapi/Authentication

Request Headers
 X-Roam-Key: <Application token>
 X-Roam-ApiVersion: 2.0.0
 X-Roam-ClientVersion: 1.0
 Content-Type: application/json
```

Request		
Parameter	**Required**	**Description**
`user_name`	Yes	User's unique identifier for logging in.
`password`	Yes	User's password.

Request Payload

```
{
    "user_name" : <username>,
    "password" : <password>
}
```

If the request is successful, you should receive a response entailing account details. Most importantly, this will return the session object, which is defined below.

Response (`session` **Object)**		
Parameter	**Required**	**Description**
`expires`	Yes	Timestamp of when the session token expires (yyyymmddhhmmss)
`session_token`	Yes	Unique session token used for every subsequent API call.

Response

```
{
  "chain_id": "860",
  ...,
  "session": {
    "expires": "20190926173844",
    "session_token": "MCM6-27651198-b49c-456a-8d4b-9794c147708a"
  },
  ...
}
```

3_2 Auth Token

Once you have successfully logged in and retrieved your Authorization token, you can utilize the Security Tokens REST API to retrieve a ***Security Token***. This security token is used to request a payment transaction with ROAMpayX.

Headers		
Parameter	**Required**	**Description**
`X-Roam-Token`	Yes	Session token from the login response. We use this to identify the API caller and its permissions.
`X-Roam-ApiVersion`	Yes	Version of the API being used (e.g., 2.0.0).
`X-Roam-ClientVersion`	Yes	Client version (can vary by application). This is set by developers (e.g., 1.0.0).
`X-Roam-TargetedUserName`	No	Used to specify a merchant or sub-merchant if requesting on their behalf (permissions restrictions may apply).
`Content-Type`	Yes	Content type (e.g., application/json).

Headers

```
Request URL:

 https://mcm.roamdata.com/wsapi/SecurityTokens

Request Headers
 X-Roam-Token: <session token from previous step>
 X-Roam-ApiVersion: 2.0.0
 X-Roam-ClientVersion: 1.0
 X-Roam-TargetedUserName: <merchant or sub-merchant username>
 Content-Type: application/json
```

Request		
Parameter	**Required**	**Description**
`token_type`	Yes	The type of token being requested. When requesting a security token, please pass the **AccessToken** value as shown in the example.
`targeted_platform`	Yes	Used to specify the platform on which you are performing the request.

Request Payload

```
{
    "token_type" : “AccessToken”,
    "targeted_platform" : “Android "
}
```

If the request is successful, you should receive a response that includes your security token. Please see below for an example of a successful response.

Response		
Parameter	**Required**	**Description**
`token`	Yes	Security token used to handshake with RPX for performing payment-related transactions.
`expires`	Yes	Timestamp of when the security token expires (yyyymmddhhmmss). **Please note: This token expires 20 seconds after the request is made.**

Response

```
{
    "token" : "MCM6-f601ecc7-95ba-4f6d-8682-1b5bcad1e435",
    "expires": "20190926173844"
}
```

3_3 Starting a Payment

Add code to start a payment in the web application. Payment transactions are initiated via a URL link that will open the RPX application installed on the device. The link URL is created using transaction information as well as the security token retrieved in Step 3_2.

Before the transaction can be started, ROAMpay X will use the auth token to login on behalf of the merchant/submerchant. However, the "login strategy" field can be added to the URL to indicate that ROAMpay X should first require the merchant to connect with a reader so that the reader's serial number can be used as the "targeted username".

a. Please note that all fields with the exception of **amount** and **authToken** are optional.
b. Definitions and enum values for the various fields can be found in the mPOS documentation. The applicable mPOS documentation can be found here.

For example:

Starting a Payment Transaction			
Parameter	**Required**	**Validation**	**Description**
authToken	Yes	length > 0	Security token used to handshake with RPX for performing payment-related transactions.
amount	Yes	Matches regex: "\d+", and amount > 0, and amount <= 10000000	Value of the transaction to be processed in cents.
tip	No	Matches regex: "\d+"	Tip to be added to the transaction, input as a value in cents.
tax	No	matches regex: "^\d{1,3}(\.\d{1,3})?$", and tax >= 0, and tax <=100	Tax to be added to the transaction, input as a percentage.
invoiceID	No	matches regex: "[a-zA-Z0-9]{0,15}"	Generated number to identify the invoice of which to associate this transaction.
transactionNotes	No	matches regex: "[\\x20-\\x7E]{0,200}"	Any additional information or notes that you'd like to associate with the transaction.
customReference Number	No	matches regex: "[-a-zA-Z0-9]{0,20}"	Custom reference identifier for the transaction. Can be used to store additional reference data with the transaction record, which can later be retrieved.
loginStrategy	No	"normal" or "readerserialnumber"	The method for which you are logging into RPX.
tenderType	No	"cash" or "keyed" or "card" or "keyedonpinpad"	If the tenderType parameter is not included then the RPX app will open with the "Choose Tender" screen. The choices offered on this screen are the tender types received from MCM that are allowed for the current merchant account.

Example request to start a payment transaction

```
var URL = "https://assets.roamdata.com/transaction/" +
"loginStrategy=readerserialnumber&" +
"authToken=" + authToken + "&" +
"amount=" + amount + "&" +
"tip=" + tip + "&" +
"tax=" + tax + "&" +
"invoiceId=" + invoiceId + "&" +
"customReferenceNumber=" + customReferenceNumber + "&" +
"transactionNotes=" + transactionNotes + "&" + //transaction notes must be URL
encoded eg. test+transaction+note or test%20transaction%20note
"tenderType=" + tenderType
;
window.open(URL);
```

3_4 Handle Transaction Response

For a Web App invocation, ROAMpay X will POST the transaction response as a JSON payload to the callback URL registered in ***Step 1 of Integration Section*** above.

In the case of a ***Native App***, ROAMpay X will send the transaction response back as a JSON string as an extra in the intent result. If a callback URL is also defined in this scenario, ROAMpay X will additionally POST the response to the URL.

Insert Card Response


```
{
  "transactions": [
    {
      "responseCode": "0",
      "authorizedAmount": 1200,
      "authCode": "PPS322",
      "transactionId": "3027582",
      "invoiceId": "2505554",
      "clerkDisplay": "APPROVED",
      "clientTransactionId": "b7c9d915-83a6-4278-b037-c216263cf1ee",
      "transactionGroupId": "0810f572-ccdd-4841-9c6e-58b1c8cfbee4",
      "sequenceNumber": "398",
      "posEntryMode": "ContactEMV",
      "cardVerificationMethod": "Signature",
      "redactedCardNumber": "472409******6261",
      "cardExpirationDate": "2111",
      "availableBalance": 0,
      "submittedAmount": {
        "currency": "USD",
        "total": 1200,
        "subtotal": 1000,
        "tax": 100,
        "discount": 0,
        "tip": 100,
        "surcharge": 0
      },
      "tokenResponseParameters": {
        "tokenResponseCode": "Unknown"
      },
      "transactionGUID": "e6707885-eb5e-45bb-a86c-14981713df16",
      "transactionResponseCode": "Approved",
      "cardType": "VISA",
```

Insert Card Response (continued…)


```
      "emvData": {
        "applicationIdentifier": "A0000000031010",
        "applicationLabel": "Visa DEBIT",
        "cryptogramType": "ARQC",
        "emvOfflineData": {
          "appCryptogram": "D0C91C71F2B777DB",
          "atc": "0031"
        }
      },
      "avsResponse": "Unknown",
      "uciFormat": "Unknown",
      "cardholderName": "SERVICES/MERCH F",
      "customerDisplay": "APPROVED",
      "batchNumber": "200218001",
      "transactionType": "CreditSale",
      "invoiceNumber": "testInvoice123",
      "customReference": "testCustomRef123",
      "transactionNotes": "test note"
    }
  ]
}
```


Example Response for a Cancellation by the User


```
{
    "transactions": [
        {
            "responseCode": "4945",
            "clerkDisplay": "Transaction Cancelled by User"
        }
    ]
}
```

Example Response for a Cancellation Due to Low Reader Battery

```
{
    "transactions": [
        {
                "responseCode": "6010",
                "clerkDisplay": "Reader Battery too low to complete payment"
        }
    ]
}
```

Example Response for an Invalid or Expired Session

```
{
    "transactions": [
        {
                "responseCode": "4995",
                "clerkDisplay": "Invalid or Expired Session"
        }
    ]
}
```

Example Response Pre-Login Errors

```
{
    "transactions": [
        {
                "responseCode": "4999",
                "clerkDisplay": "Unknown Error"
        }
    ]
}
```

Example to Retrieve a Transaction Response JSON via an Intent

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
      String response = data.getStringExtra("INTENTEXTRA_APPLINKRESPONSEJSON");
    if (resultCode == Activity.RESULT_OK) {
            // transaction was successfully processed

    } else if (resultCode == Activity.RESULT_CANCELED) {
            // transaction was not successfully processed
      }
}
```

Example response for Invalid Tender Type

```
{
      "transactions" : [
      {
                  "responseCode": 4993
                  "clerkDisplay": "Invalid Tender type",
      }
      ]
}
```

Example Response: Partial Auth, Decline and Reversal (1/3)


```
{
    "transactions": [
      {
        "responseCode": "0",
        "authorizedAmount": 1000,
        "authCode": "PPS323",
        "transactionId": "3027583",
        "invoiceId": "2505555",
        "clerkDisplay": "Balance due: $1500.00",
        "clientTransactionId": "91b09fd5-95da-413d-a37b-8b1a1c4aeb77",
        "transactionGroupId": "f9bda6c5-d718-4f73-ad09-7b51e4ca99b1",
        "sequenceNumber": "399",
        "posEntryMode": "ContactEMV",
        "cardVerificationMethod": "Signature",
        "redactedCardNumber": "472409******6261",
        "cardExpirationDate": "2111",
        "availableBalance": 0,
        "submittedAmount": {
          "currency": "USD",
          "total": 151000,
          "subtotal": 151000,
          "tax": 0,
          "discount": 0,
          "tip": 0,
          "surcharge": 0
        },
        "tokenResponseParameters": {
          "tokenResponseCode": "Unknown"
        },
        "transactionGUID": "93f2aae9-5137-4470-9f4e-bdb2cca396a0",
        "transactionResponseCode": "Approved",
        "cardType": "VISA",
        "emvData": {
          "applicationIdentifier": "A0000000031010",
          "applicationLabel": "Visa DEBIT",
          "cryptogramType": "ARQC",
          "emvOfflineData": {
            "appCryptogram": "CDC8A583090EE042",
            "atc": "0032"
          }
        },
        "avsResponse": "Unknown",
```

Example Response: Partial Auth, Decline and Reversal (2/3)


```
    "uciFormat": "Unknown",
    "cardholderName": "SERVICES/MERCH F",
    "customerDisplay": "Balance due: $1500.00",
    "batchNumber": "200218001",
    "transactionType": "CreditSale",
    "invoiceNumber": "testInvoice123",
    "customReference": "testCustomRef123",
    "transactionNotes": "test note"
  },
  {
    "responseCode": "0",
    "authorizedAmount": 0,
    "clientTransactionId": "0e7d7029-4930-4e58-a259-f3d4a61450f3",
    "posEntryMode": "Unknown",
    "cardVerificationMethod": "None",
    "redactedCardNumber": "",
    "cardExpirationDate": "",
    "availableBalance": 0,
    "submittedAmount": {
      "currency": "USD",
      "total": 150000,
      "subtotal": 151000,
      "tax": 0,
      "discount": 0,
      "tip": 0,
      "surcharge": 0
    },
    "tokenResponseParameters": {
      "tokenResponseCode": "Unknown"
    },
    "transactionResponseCode": "Declined",
    "cardType": "Unknown",
    "emvData": {
      "emvOfflineData": {}
    },
    "avsResponse": "Unknown",
    "uciFormat": "Unknown",
    "transactionType": "Unknown",
    "invoiceNumber": "testInvoice123",
    "customReference": "testCustomRef123",
    "transactionNotes": "test note"
  },
```

Example Response: Partial Auth, Decline and Reversal (3/3)


```
    {
      "responseCode": "0",
      "authorizedAmount": 1000,
      "authCode": "PPS323",
      "transactionId": "3027584",
      "invoiceId": "2505555",
      "clerkDisplay": "APPROVED",
      "clientTransactionId": "1996b5bd-2e0c-4388-8701-623a4482a7af",
      "transactionGroupId": "3a8599cc-1acb-4c10-9eea-011c9657536d",
      "sequenceNumber": "400",
      "posEntryMode": "Keyed",
      "cardVerificationMethod": "None",
      "redactedCardNumber": "472409XXXXXX6261",
      "cardExpirationDate": "2111",
      "availableBalance": 0,
      "submittedAmount": {
        "currency": "USD",
        "total": 1000
      },
      "tokenResponseParameters": {
        "tokenResponseCode": "Unknown"
      },
      "transactionGUID": "a101d2c7-7137-477d-aefe-aaae55e03e4d",
      "transactionResponseCode": "Approved",
      "cardType": "VISA",
      "emvData": {
        "emvOfflineData": {}
      },
      "avsResponse": "Unknown",
      "uciFormat": "Unknown",
      "customerDisplay": "APPROVED",
      "batchNumber": "200218001",
      "transactionType": "CreditSaleVoid",
      "invoiceNumber": "testInvoice123",
      "customReference": "testCustomRef123",
      "transactionNotes": "test note"
    },
    {
      "responseCode": "4945",
      "clerkDisplay": "Transaction Cancelled by User"
    }
  ]
}
```

3_5 Error Messages

If the validation fails, ROAMpay X will display an alert with an appropriate title and message mentioned above and the user will be brought back to the calling app upon dismissing the dialog.

Scenario	**Error Message**
If the account was not set up properly with API support	**Title:** Invalid Configuration **Message:** The callback URL for the user has not been configured. Please contact ROAMsupport **Button:** Cancel Transaction
If the version of Google Play Services is too low.	**Title:** Device Configuration Check Failed **Message:** Payment transaction cannot continue. An update to Google Play Services is required. Tap on the system notification or visit Google Play for details. **Button:** Cancel Transaction
If the Android device boot loader is not locked, or If the Android device does not have a clean factory ROM, or If there's a network error, or If the device integrity check fails for some other reason.	**Title:** Device Integrity Check Failed **Message:** Could not verify device security and integrity **Button:** Cancel Transaction
If the requested tender type is not permitted for the current merchant account.	**Title:** Cannot perform transaction. **Message:** The requested tender type is not allowed for this account. Please try a different tender type. **Button:** Cancel Transaction