ROAMpay X Universal Link Integration Guide

iOS ONLY

Version 2.0
April 2021

Revision History

Version Number	Revision Date	Revision Description
v1.0	1/2020	Initial release.
v2.0	4/2021	Included multiple new response examples for Web App, as well as included support for tenderType.

Table of Contents

1 Introduction

The objective of the document is to outline the different steps involved in getting started with payment processing with ROAMpay X iOS from your web browser or native iOS application. The universal link feature in ROAMpay X lets web and native applications open ROAMpay X to process in-person payments using supported Ingenico card readers.

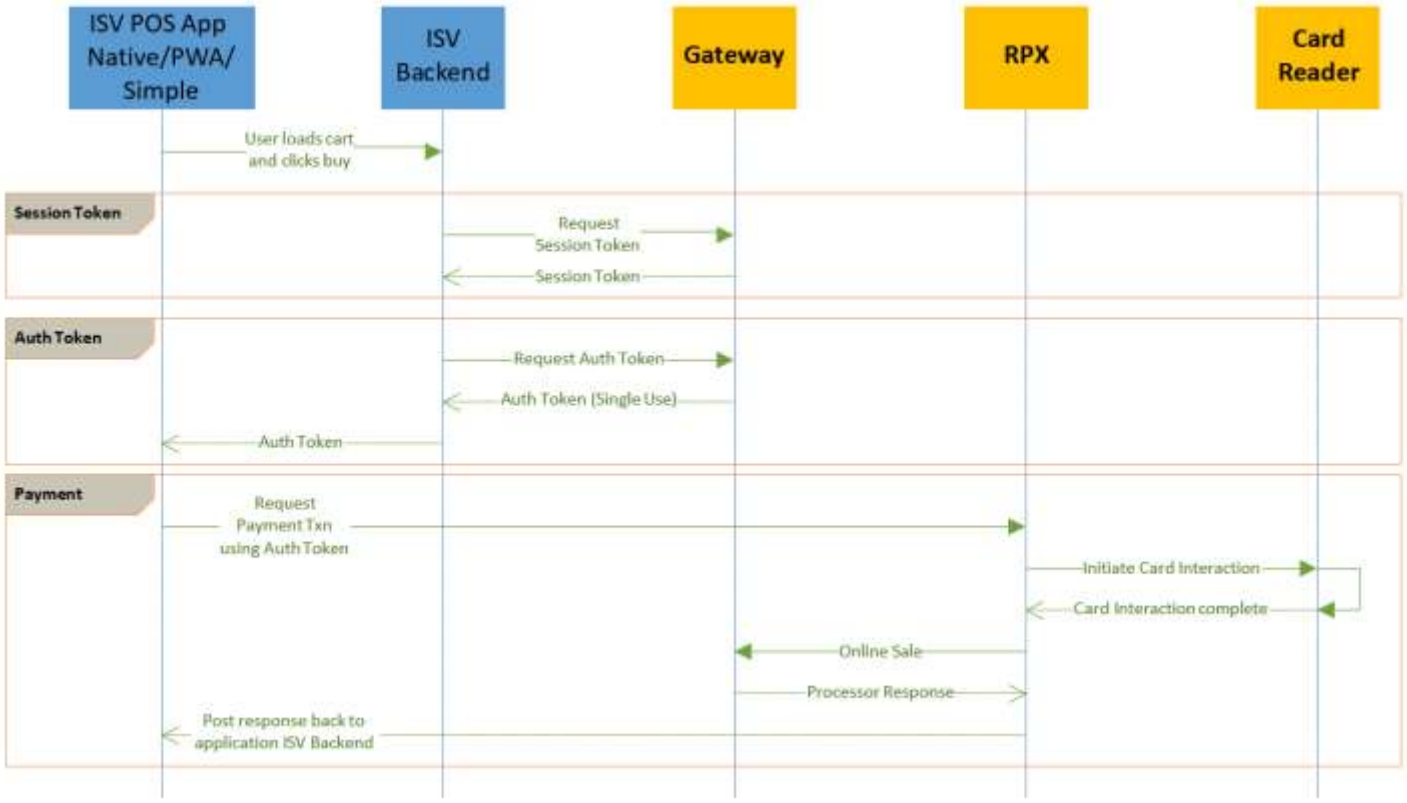
1_1 Requirements

- iOS 9.0+
- Universal Link
    - Universal Link is an iOS feature that allows seamless content linking through both a browser and within an application. Universal Links are standard web links used by iOS to see if an installed application is registered for that domain. If so, the app is launched immediately without ever loading the web page.

    - When users install the ROAMPayX app, iOS checks a file(Apple App Site Association) that's been uploaded to our web server to make sure that our website allows your app to open ROAMPayX on its behalf. Only Ingenico can create and upload this file, so the association of our app with your app is secure. For more information, visit the Apple developer documentation.

1_2 Workflow

Please see the diagram below for a high-level overview of the deep link 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 user account. 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.

2. For production build, Install ROAMpay X5 from the Appstore .
   a. For Debug build, Install the ROAMpay™ X5(Debug) app build from Fabric Beta app. Users will have to accept the invitation sent by Fabric to install the Fabric Beta app.

3. Add code to log into the Ingenico system and retrieve an authorization token.
   a. Using the Ingenico Authorization REST API, log into the system with the username and password to retrieve a session token. This login request can be from any account (including ISO) within the organization structure.

3 Logging in

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

```
{
    "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_1 Authorization

Once you have successfully logged in and retrieved your session 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

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

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.**

Request

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

4 Starting a Payment

Add code to start a payment in the web application. Payment transactions are initiated via URL links 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 2_2.

a. Please note that all fields with the exception of **amount**, **authToken** and **hostURL** are optional.
    i. The hostURL parameter is mandatory for iOS as RPX will use this to call the ISV's application after completing the transaction.
b. Amount and tip are expected to be input as cent values, and tax is expected to be input as a percentage.
c. 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:

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.

Starting a Payment Transaction			
Parameter	**Required**	**Validation**	**Description**
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.

```
NSString *urlString = [NSString
stringWithFormat:@"https://assets.roamdata.com/transaction/
                        amount=%@&
                        tip=%@&
                        tax=%@&
                        authToken=%@&
                        invoiceId=%@&
                        customReferenceNumber=%@&
                        transactionNotes=%@&
                        hostURL=%@",
                        &loginstrategy=ReaderSerialNumber&tenderType=card,

                        amount,tip,tax,authToken,invoiceid,customReferenceNumber,c
ustomReferenceNumber, @"RPXUniversalLinkNativeTestApp://"];
NSURL *url = [NSURL URLWithString:urlString];
[[UIApplication sharedApplication] openURL:url];
```

4_1_1 hostURL on Native iOS Applications

The ISV's native application should define the Custom URL, and this Custom URL should be sent as hostURL. This hostURL must be a unique and end with "://", as per the Apple developer documentation. For more information, click here.

4_1_2 hostURL on Web Applications

For web applications, hostURL should be a proper http:// or https:// . ROAMPay X will open the Safari to load this URL after completing the transaction.

4_2 Response (Web App)

RPX will send the data to the callback URL and/or hostURL as a JSON payload (example shown below), which will need to be parsed by the app The ISV's web app will receive the transaction JSON response via callback URL and the native app will receive the transaction JSON response via host URL and callback URL if it exists.

Swipe Card Response


```
{
  "transactions" : [
    {
      "emvData" : {
        "emvOfflineData" : {
        }
      },
      "batchNumber" : "200326001",
      "tokenResponseParameters" : {

      },
      "transactionGUID" : "fb1514ff-412a-4fea-95c1-b0d0e350c62b",
      "submittedAmount" : {
        "currency" : "USD",
        "total" : 150000,
        "subTotal" : 152000
      },
      "authCode" : "PPS041",
      "cardExpirationDate" : "2412",
      "transactionGroupID" : "40596CC8-F21E-4249-A91B-C21D5B4A5EE4",
      "invoiceID" : "2516504",
      "transactionResponseCode" : "Approved",
      "sequenceNumber" : "3229",
      "authorizedAmount" : 150000,
      "cardType" : "MasterCard",
      "transactionType" : "CreditSale",
      "posEntryMode" : "MagStripe",
      "cardVerificationMethod" : "Signature",
      "clientTransactionID" : "62A589C6-BCC6-491A-BDB5-1391A8A4B690",
      "customerDisplay" : "APPROVED",
      "transactionID" : "3046615",
      "clerkDisplay" : "APPROVED"
    }
  ]
}
```

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 response for Invalid Tender Type

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

4_3 Response (Native App)

For native applications, ROAMPayX will return the JSON response as base64 encoded string. Therefore, ISV native apps should implement the openURL method in AppDelegate class to receive the base64 encoded string transaction response. Base 64 encoded string can be converted to JSON string as shown below.

Converting Base 64 to JSON

```
- (BOOL)application:(UIApplication *)app openURL:(NSURL *)url
    options:(NSDictionary<UIApplicationOpenURLOptionsKey, id> *)options{
    NSString *transactionResponseBase64EncodedString = [[[url query]
    componentsSeparatedByString:@"transactionResponse="] lastObject];
    NSData *transactionResponse = [[NSData alloc]
    initWithBase64EncodedString:transactionResponseBase64EncodedString
    options:0];
    NSString *transactionResponseJSONString = [[NSString alloc]
    initWithData:transactionResponse encoding:NSUTF8StringEncoding];
    return YES;
}
```

ROAMpayX will open the hostURL, bringing the ISV application to the foreground and RPX will enter the background.