History

Version	Date	Author	Description/Comments
1.0	07/07/15	Venkat Saddala	Initial Draft

Table of Contents

Using ApplePay API’s with Authorize.Net

Apple Pay allows you to use your Authorize.Net account to securely accept and process in-app payments from customers with the new iPhone 6 and iPhone 6 Plus. Apple Pay uses the tokenization solutions developed by VISA, MasterCard and American Express.

When a customer checks out in our app, Apple Pay will tokenize the credit card information from the device and use that token for the transaction. This means customer doesn’t have to enter their credit card information, because Apple Pay will pass the tokenized card information to Authorize.Net.

This document is intended to be used by Kony (JS Bindings) developers who want to offer Apple Pay in their app, and use Authorize.Net to process the payments by integrating with the AIM XML API.

1. API Endpoints

Making a REST API call with Authorize.Net requires you to create XML Service in Kony application. Following are the service end point details.

Request Method: POST

Sandbox URL: https://apitest.authorize.net/xml/v1/request.api

Production URL: https://api.authorize.net/xml/v1/request.api

XML Content-Type: text/xml

JSON Content-Type: application/json

2. Creating an Apple Pay Transaction Payload

For making Apple Pay transaction with Authorize.net payment gateway requires creating a XML payload. Following is the sample XML payload.

XML Payload Format

```
<?xml version="1.0" encoding="utf-8"?>
<createTransactionRequest xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="AnetApi/xml/v1/schema/AnetApiSchema.xsd">
        <merchantAuthentication>
                <name>$nameStr</name>
                <fingerPrint>
                        <hashValue>$!hashValueStr</hashValue>
                        <sequence>$!seq</sequence>
                        <timestamp>$!stamp</timestamp>
                </fingerPrint>
        </merchantAuthentication>
        <transactionRequest>
                <transactionType>authCaptureTransaction</transactionType>
                <amount>!$amount</amount>
                <payment>
                        opaqueData>
                                <dataDescriptor> COMMON.APPLE.INAPP.PAYMENT  </dataDescriptor>
                                <dataValue>$dataValueStr</dataValue>
                        </opaqueData>
                </payment>
                <order></order>
                <lineItems></lineItems>
                <customer>
                        <email>Venkat.saddala.kony.com</email>
                </customer>
                <billTo>
                        <firstName>Venkat</firstName>
                        <lastName>Saddala</lastName>
                        <address>7380 West Sand Lake #390 Orlando</address>
                        <city>Florida</city>
                        <state>ORLANDO</state>
                        <zip>32819</zip>
                        <country>United States</country>
                        <phoneNumber>8121-013945</phoneNumber>
                </billTo>
                <shipTo>
                        <firstName>Venkat</firstName>
                        <lastName>Saddala</lastName>
                        <address>7380 West Sand Lake #390 Orlando</address>
                        <city>Florida</city>
                        <state>ORLANDO</state>
                        <zip>32819</zip>
                        <country>United States</country>
                </shipTo>
                <retail>
                        <marketType>0</marketType>
                        <deviceType>7</deviceType>
                </retail>
                <transactionSettings></transactionSettings>
                <userFields></userFields>
        </transactionRequest>
</createTransactionRequest>
```

2.1. Merchant Authentication:

```
        <merchantAuthentication>
                <name>$nameStr</name>
                <fingerPrint>
                        <hashValue>$!hashValueStr</hashValue>
                        <sequence>$!seq</sequence>
                        <timestamp>$!stamp</timestamp>
                </fingerPrint>
        </merchantAuthentication>
```

All calls to the Authorize.Net API require merchant authentication. Merchant Authentication element contains information required to authenticate yourself with Authorize.net. To complete merchant authentication, you need to generate finger print hash value.

2.1.1. Generating the Unique Transaction Fingerprint

Transaction authentication for Server Integration Method is a transaction fingerprint, or a hash of merchant- and transaction-specific information using the HMAC-MD5 hashing algorithm (Hash-based Message Authentication Code) (MD5 RFC 1321 with a 128-bit hash value). The HMAC-MD5 algorithm is used only for generating the unique transaction fingerprint. The transaction fingerprint must be generated for each transaction by a server-side script on the merchant's web server and inserted into the transaction request. The payment gateway uses the same mutually exclusive merchant information to decrypt the transaction fingerprint and authenticate the transaction.

You can develop a script for generating a fingerprint in two ways:

- By using the API field information to customize your script.
- By using a free Authorize.Net sample code available on the Developer Center here.

In our Kony sample app, we are using Java code to generate fingerprint in XML service Pre-Processor. You need to replace the values generated in the Pre-Processor in the XML payload before sending the payment request to the Authorize.net to process.

2.2. Transaction Type

Authorize.net supports only 'authCaptureTransaction' transaction types for Apple Pay.

2.3. Opaque Data

```
<opaqueData>
        <dataDescriptor> COMMON.APPLE.INAPP.PAYMENT </dataDescriptor>
        <dataValue>$dataValueStr</dataValue>
</opaqueData>
```

2.3.1 dataDescriptor:

Meta data used to specify how the request should be processed. The value of dataDescriptor is based on the source of the opaqueData dataValue. For Apple Pay, the value is COMMON.APPLE.INAPP.PAYMENT.

2.3.2 dataValue:

8192 characters. Base-64 encoded data that contains encrypted payment data. The payment gateway expects the encrypted payment data and Meta data for the encryption keys.

2.4 Other XML payload Elements

For the rest of the XML payload elements, please refer to the 'Apple Pay Transactions' section at.

https://developer.authorize.net/api/reference/#apple-pay-transactions

3. Enabling Apple Pay on Authorize.net account

There are two steps required to sign up for Apple Pay with Authorize.Net.

1. You must enable the service in the Merchant Interface.
2. You must generate a CSR file to upload to Apple.

3.1 Enabling Apple Pay in the Merchant Interface

1. An Account Owner must log into the Merchant Interface at https://account.authorize.net. (For Sandbox account, https://sandbox.authorize.net)
2. Click Account from the main toolbar.
3. Click Digital Payment Solutions from the menu on the left.
4. Click Sign Up in the Apple Pay section.
5. The account must meet the below requirements for signing up:

   API Login ID and Transaction Key must have been generated (If you or your developer don't remember your Transaction Key, you must generate a new one. If you do generate a new Transaction Key, be sure to update the rest of your solutions with the new ID.)

   Processor must be supported (Chase Paymentech, Global Payments and TSYS are currently supported, with First Data coming in early 2015.)

6. Enter the Apple Merchant ID that you or your developer obtains in the "Certificates, Identifiers, & Profiles" area of the Member Centre at Apple.

3.2 Generating the CSR File

A Certificate Signing Request (CSR) file must be submitted to Apple to receive a necessary payment entitlement certificate.

To generate the CSR file:

1. Log into the Merchant Interface at https://account.authorize.net. (For Sandbox account, https://sandbox.authorize.net/)
2. Click Account from the main toolbar.
3. Click Digital Payment Solutions from the menu on the left.
4. Click Apple Pay.
5. Enter your Apple Merchant ID in the field provided.
6. Click the Generate Apple Input File button.

3.3 Submit CSR File to Apple to create Merchant ID

Submit the CSR file to Apple to get the required payment entitlement certificate.

1. Navigate to the Certificates, Identifiers, & Profiles area of the Member Centre at the Apple World Wide Developer Relations (WWDR) web site.

2. In the Register Merchant IDs section, click Continue. Your Merchant ID is in the Identifier field.
3. Click Done.
4. In the Merchant ID page, click Edit.
5. In the iOS Merchant ID Settings page, click Create Certificate.
6. Follow the instructions to submit the CSR, you created at Authorize.net.

4. Settings to avoid Error Code – 153

For Apple Pay transactions with Authorize.net, service returns you with error code 153 for all types of unsuccessful transactions. Following are the important settings you need to make before sending the request to avoid error code 153.

- Cannot include with card number or expiration date
  - You should not include any data related to card number or expiry date in the xml payload as part of payment request.
- Cannot include track data
  - You should not include any data related order track as part of payment request.
- Must be ecommerce transaction, please make sure your gateway is setup as CNP (card Not Present)
  - Accounts which you create are CNP by default.
- Must be auth or authcapture type of transaction
  - XML data which you are trying to send should be of type 'authcapture' transaction.
- Include 3DS data
  - This is related to apple pay app's payment request object setting. You can set in the app code as follows.
  - request.merchantCapabilities = PKMerchantCapability3DS; //PKMerchantCapabilityEMV;

Note: When we tried with 'PKMerchantCapabilityEMV' capability, we were receieving error code-153

5. Kony Apple Pay sample app resources links

Apple Pay JS Bindings Project:

https://konyone.sharepoint.com/sites/Kony/PlatformQAMain/_layouts/15/guestaccess.aspx?guestaccesstoken=L9%2bpa4vWCrBIMP3NaJIBTTbG4HwYnOPVPzYC8B2nSxs%3d&docid=101c0cd53417f4881ad62dcc76a7367dc

XML Service Pre Processor (Java) Project

https://konyone.sharepoint.com/sites/Kony/PlatformQAMain/_layouts/15/guestaccess.aspx?guestaccesstoken=gISXOtWGdl%2b9c0m/h4%2bpNYSpKBVJkjkcnHfpndqEUEc%3d&docid=14724228213c54e0aa4279a2b92b8e6c5