Getting Started with Apple Pay on the Authorize.Net Platform

September 2014

Authorize.Net

Authorize.Net Developer Support
http://developer.authorize.net
Authorize.Net LLC 082007 Ver.2.0

Authorize.Net LLC (“Authorize.Net”) has made efforts to ensure the accuracy and completeness of the information in this document. However, Authorize.Net disclaims all representations, warranties and conditions, whether express or implied, arising by statute, operation of law, usage of trade, course of dealing or otherwise, with respect to the information contained herein. Authorize.Net assumes no liability to any party for any loss or damage, whether direct, indirect, incidental, consequential, special or exemplary, with respect to (a) the information; and/or (b) the evaluation, application or use of any product or service described herein.

Authorize.Net disclaims any and all representation that its products or services do not infringe upon any existing or future intellectual property rights. Authorize.Net owns and retains all right, title and interest in and to the Authorize.Net intellectual property, including without limitation, its patents, marks, copyrights and technology associated with the Authorize.Net services. No title or ownership of any of the foregoing is granted or otherwise transferred hereunder. Authorize.Net reserves the right to make changes to any information herein without further notice.

Authorize.Net Trademarks

Advanced Fraud Detection Suite™

Authorize.Net®

Authorize.Net Your Gateway to IP Transactions™

Authorize.Net Verified Merchant Seal™

Automated Recurring Billing™

eCheck.Net®

FraudScreen.Net®

Contents

Getting Started with Apple Pay On the Authorize.Net Platform

Recent Revisions

Release	Changes
September 2014	First release of this guide.

Audience and Purpose

This document is intended to be used by Authorize.Net merchants who want to offer Apple Pay in their iOS application and use Authorize.Net to process the payments. Merchants who do not have an Authorize.Net account must obtain a merchant-acquiring bank account and separately sign up for an Authorize.Net account before using this guide.

Important

Apple Pay relies on payment network tokenization.

You can sign up for Apple Pay only if the following statements are true:

- Your processor supports payment network tokenization.
- Authorize.Net supports the tokenization interface with your processor.

If the preceding statements are not true, you must take one of the following actions before you can sign up for Apple Pay:

- Obtain a new merchant account with a processor that supports payment network tokenization.
- Wait until your processor supports payment network tokenization.

Note

Authorize.net makes it easier to get a merchant account with a preferred processor. Navigate to http://www.authorize.net/solutions/merchantsolutions/merchantservices/applepay/ to get an account.

Related Documentation

Authorize.Net SDK Developer Reference for Apple Pay

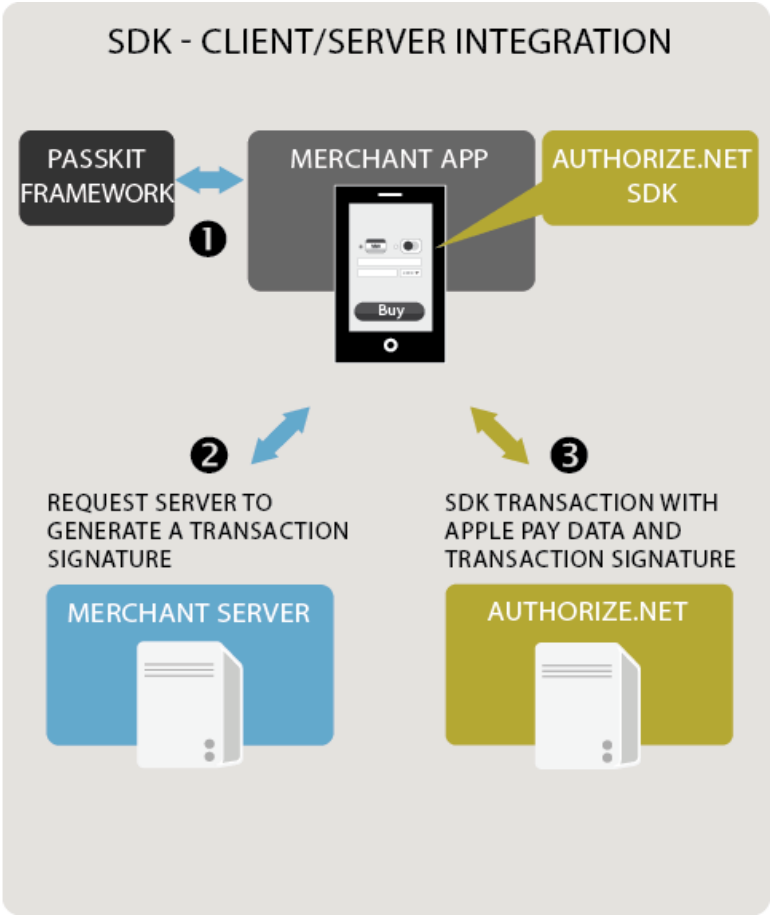
Authorize.Net AIM and AIM XML Supplement for Apple Pay

How Apple Pay Works

The merchant's iOS app uses the Passkit Framework to request encrypted payment data from Apple. Apple then returns the encrypted payment data. At that point, there are two methods, the SDK method and the API method, shown below.

SDK Method

Figure 1 SDK Method



Note

The merchant must sign up for this free service in the merchant interface. For more information, see "Signing Up for the Service," page 8.

1. The merchant's iOS app interacts with the Apple Passkit Framework to obtain encrypted payment data.
2. The app creates a transaction fingerprint using the merchant's server.
3. The app sends the transactions to Authorize.Net using the Authorize.Net SDK.
4. Authorize.Net decrypts the data and runs a transaction.

Use the *Authorize.Net SDK Developer Reference for Apple Pay* to download the Authorize.Net SDK and build the required payment capacity into your app. The SDK supports two types of credit card transaction types: Authorization and Capture and Authorization Only. Choose the one relevant to your business and which is supported by your processor.

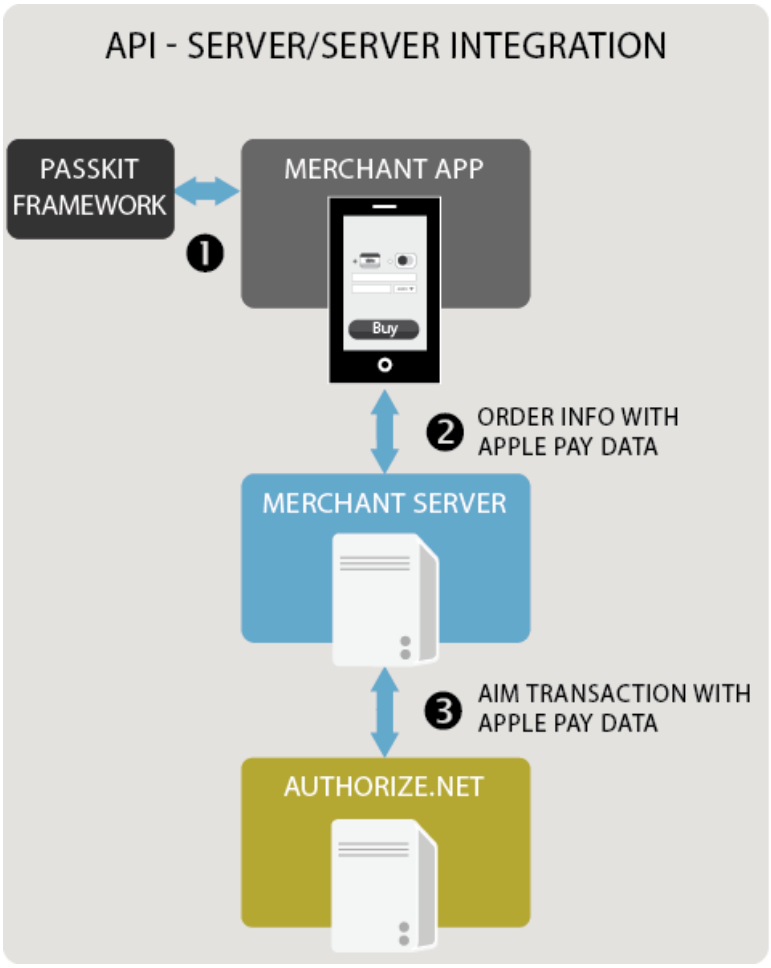
The subsequent transactions, such as Capture, Refund, and Void, should be done using the Merchant Interface or the Authorize.Net payment APIs. Since they are not relevant to the app on a consumer device, they are not supported by the SDK. Please note that if you have chosen Authorization Only transactions in the SDK, you must complete the subsequent transactions for you to receive the funds.

Note

If you are using a sandbox account on Authorize.Net to test, you must connect to the Apple developer system and not to the Apple production systems.

API Method

Figure 2 API Method



Note

The merchant must sign up for this free service, in the merchant interface. For more information, see "Signing Up for the Service," page 8.

1. The merchant's app interacts with the Apple Passkit Framework to obtain encrypted payment data.
2. The merchant's app forwards the order information with encrypted payment data to the merchant's web server.
3. The merchant's web server forwards the encrypted payment data to Authorize.Net using the AIM or AIM XML API.
4. Authorize.Net decrypts the data and runs a transaction.

The API method may be useful to you if you want to centralize all your payment requests through your server. In this case, you will need to extract the encrypted payment data in your app and send it over to your server. Your server would send in this payment data

instead of the PAN information. The transaction is submitted using the AIM API (XML or NVP). The encrypted data (base64 encoded) and a format descriptor are sent to create this transaction. Authorize.Net extracts the relevant information from this payment data, processes the transaction appropriately, and sends the response back to your server.

For more information, see the *Authorize.Net AIM and AIM XML Supplement for Apple Pay*.

Creating a Transaction Key

If you have not already done so, you must create a transaction key. This key is used to create a transaction-specific signature at your server. The signature is cross-checked by Authorize.Net servers when a transaction request is received, ensuring that no transaction is initiated by the consumer without your server's approval.

Please ensure that your transaction key is stored securely on your merchant server. It should never be stored within your app code on the device. If the key is compromised, you should generate a new key.

To create a transaction key:

Step 1 In the Authorize.Net Merchant Interface, navigate to **Account > Settings > Security Settings > General Security Settings > API Login and Transaction Key**.

Step 2 Enter the answer to the secret questions by entering it into the **Secret Answer** field.

Step 3 Click **Submit**.

Signing Up for the Service

The Apple Pay solution uses payment network tokenization. You can sign up for this solution only if your payment processor supports tokenization. If your processor does not support payment network tokenization or if Authorize.Net does not support your payment processor's tokenization interface, you will not be able to sign up for this payment solution.

To sign up for Apple Pay:

Step 1 Log in to the Authorize.Net Merchant Interface.

Step 2 Navigate to the **Digital Payment Solutions** section.

Step 3 In the Apple Pay section, click **Sign Up**.

Obtaining an Apple Merchant ID

You must obtain an Apple Merchant ID before you can generate the Certificate Signing Request (CSR) that Apple requires.

To obtain an Apple Merchant ID:

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

Step 2 In the Register Merchant IDs section, click **Continue**. Your Merchant ID is in the **Identifier** field.

Step 3 Click **Done**.

Generating the CSR

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

To generate the CSR:

Step 1 Log in to the Authorize.Net Merchant Interface.

Step 2 Navigate to **Home > Digital Payments Solutions > Apple Pay**

Step 3 Enter your Apple Merchant ID in the **Apple Merchant ID** field.

The Apple Merchant ID that you enter should be identical to the one that you created at the Apple site. Any difference will cause Authorize.Net to be unable to decrypt the payment data.

Step 4 Click the **Generate Apple CSR** button.

Submitting the CSR File to Apple

You must submit the CSR to Apple to get the required payment entitlement certificate.

To submit the CSR file to Apple:

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

Step 2 In the Merchant ID page, click **Edit**.

Step 3 In the iOS Merchant ID Settings page, click **Create Certificate**.

Step 4 Follow the instructions to submit the CSR .

Identifying Apple Pay Transactions in the Merchant Interface

You can identify Apple Pay transactions in the Transaction Detail view of the Authorize.Net Merchant Interface.

To identify an Apple Pay transaction:

Step 1 Log in to the Authorize.Net Merchant Interface and navigate to **Reports > Transaction Detail**.

Step 2 Choose an **Item Type** and a **Date** from their respective pull-down menus and then click **Run Report**.

Step 3 Choose a transaction by clicking its **Trans ID**.

The Transaction Details lightbox appears. Apple Pay information is contained in the **Tokenization Information** section.