Apple Pay Integration Guide (In App / On the Web)

March 2018

Contents

Overview

First Data's RESTful API allows developers to quickly enable secure and convenient payments in their payment applications. The API handles all of the tokenization needed to protect customers' transactions.

This documentation refers to both the In-App and On the Web Apple Pay (In App) integrations.

Target Audience

The target audience for this document is a developer who wants to use Apple Pay in their payment application, both for in app payments and for on the web payments.

Minimum Technical Requirements

Developers wishing to use the First Data Apple Pay (In App) sample application will need the following software and hardware:

- Xcode 6.3 and above
- iOS 8.0 and above
- A physical device or an emulator to use for developing and testing. Refer to Apple Pay for compatible devices.

Related Documents and Resources

The following First Data documents and resources provide supporting information for this document:

- Developer Portal (https://developer.payeezy.com/) - This portal is where developers register and are boarded to First Data's RESTful API. It also contains specifications for all supported APIs, sample requests and responses, and development guides. The following items are especially pertinent to readers of this guide:
    - Apple Pay Transaction API
    - Getting Started
    - FAQs
- First Data Apple Pay Example – This sample application shows how to integrate the First Data APIs to support the Apple Pay feature.

Third Party Resources

In addition to the First Data resources, the following third party documents and resources provide supporting information for working with the First Data APIs and sample Apple Pay application.

Apple Pay Resources

Developers should refer to the Apple Pay documentation, available at the Apple Pay Developer site, particularly the following:

- Getting Started with Apple Pay
- iOS Human Interface Guidelines: Apple Pay
- Apple Pay Programming Guide
- PKPaymentAuthorizationViewController Class Reference
- Apple Pay Within Apps (for In-App integrations) or Apple Pay JS (for Apple Pay On the Web)
- Apple Pay Identity Guidelines

Download and Build Resources

- GitHub (https://github.com/) – Recommended for First Data sample application code downloads
- Gradle (http://gradle.org/) – For builds, the standard Gradle build can be used.

Pre-Requisites (In App and On the Web)

Before integrating with First Data's APIs, developers should first:

- Review the prerequisites described in the First Data Getting Started Guide.
- Complete registration and certification to the APIs at the Developer Portal, if necessary.
- Download a Certificate Signing Request (CSR) from the **My Merchants** page in the developer's Developer Portal account.

To continue with an In App integration, see In-App Integration, below. To continue with an Apple Pay On the Web integration, skip to On the Web Integration.

In-App Integration

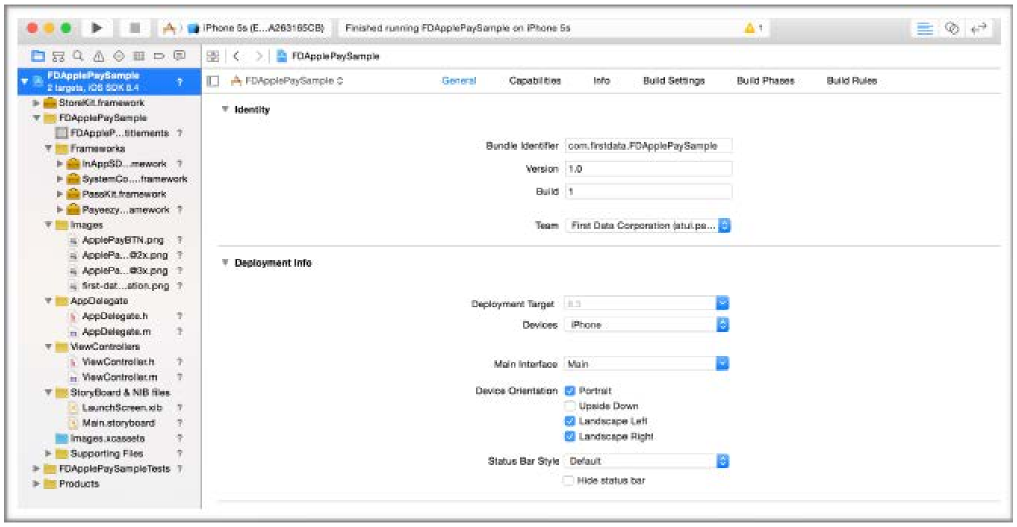
This section describes the how to integrate with an in-app payment page.

Apple Pay Configurations and Settings

Clone (desktop) or download the appropriate First Data sample Apple Pay library and example from the First Data Apple Pay github (https://github.com/payeezy/payeezy_apple_pay).

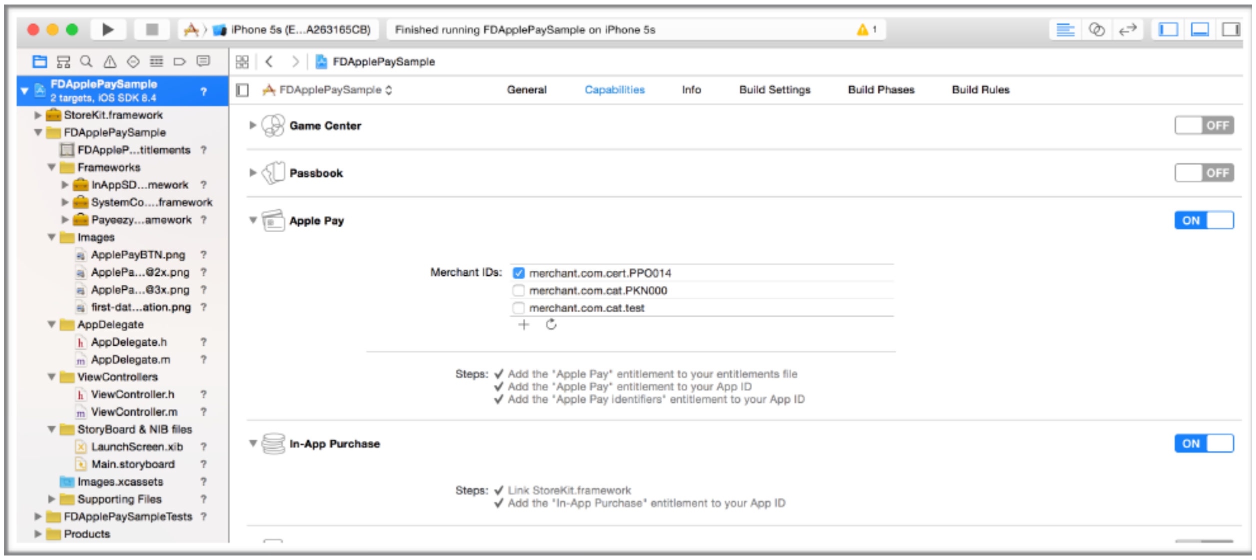
Once you have downloaded or cloned the First Data library and example, open the project in Xcode.

Click **Project**, and then select your team. (The code may not compile.)



Bundle Identifier: com.firstdata.FDApplePaySample
Version: 1.0
Build: 1
Team: First Data Corporation (atul.pa...
Deployment Target: 8.3
Devices: iPhone
Main Interface: Main
Device Orientation: ☑ Portrait ☐ Upside Down ☑ Landscape Left ☑ Landscape Right
Status Bar Style: Default
☐ Hide status bar

Under **Capabilities**, enable **Apple Pay** and **In-app Purchase**. The code may not compile as it requires a Merchant Certificate and Apple-Pay provision for your bundle-indentifier.



Game Center OFF

Passbook OFF

Apple Pay ON

Merchant IDs: [✓] merchant.com.cert.PPO014
[ ] merchant.com.cat.PKN000
[ ] merchant.com.cat.test

Steps: ✓ Add the "Apple Pay" entitlement to your entitlements file
✓ Add the "Apple Pay" entitlement to your App ID
✓ Add the "Apple Pay identifiers" entitlement to your App ID

In-App Purchase ON

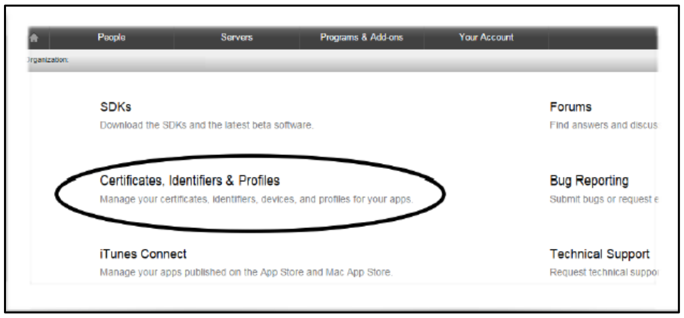
Steps: ✓ Link StoreKit.framework
✓ Add the "In-App Purchase" entitlement to your App ID

Certificates, Identifiers and Profiles

Use the CSR to generate a Merchant certificate.

To register a Merchant ID:

1. Sign in to developer.apple.com
2. In the Member Center, click Certificates, Identifiers & Profiles, as shown below.



SDKs
Download the SDKs and the latest beta software.

Certificates, Identifiers & Profiles
Manage your certificates, identifiers, devices, and profiles for your apps.

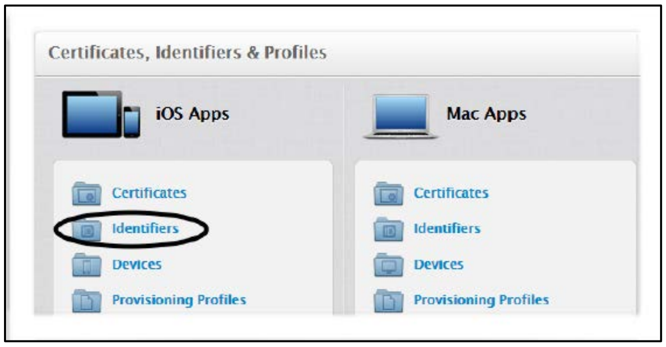
iTunes Connect
Manage your apps published on the App Store and Mac App Store.

Forums
Find answers and discus

Bug Reporting
Submit bugs or request e

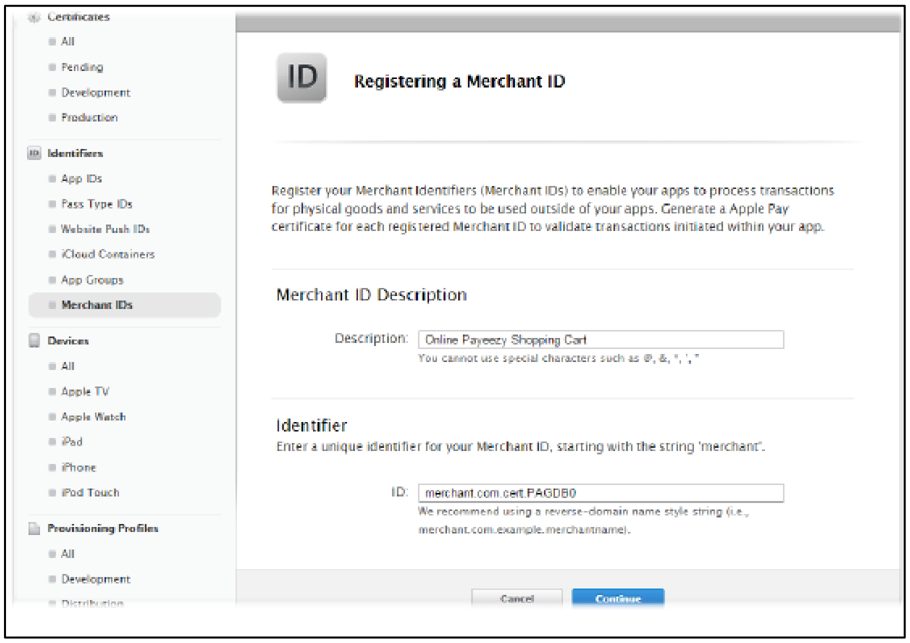
Technical Support
Request technical suppo

3. Click Identifiers, as shown below.



4. Click Merchant IDs, and then enter the Merchant Description/Name and Identifier, as shown below.

   Note: The Merchant ID must match the App Lable from the My Merchants page in the First Data Developer Portal.



Registering a Merchant ID

Register your Merchant Identifiers (Merchant IDs) to enable your apps to process transactions for physical goods and services to be used outside of your apps. Generate a Apple Pay certificate for each registered Merchant ID to validate transactions initiated within your app.

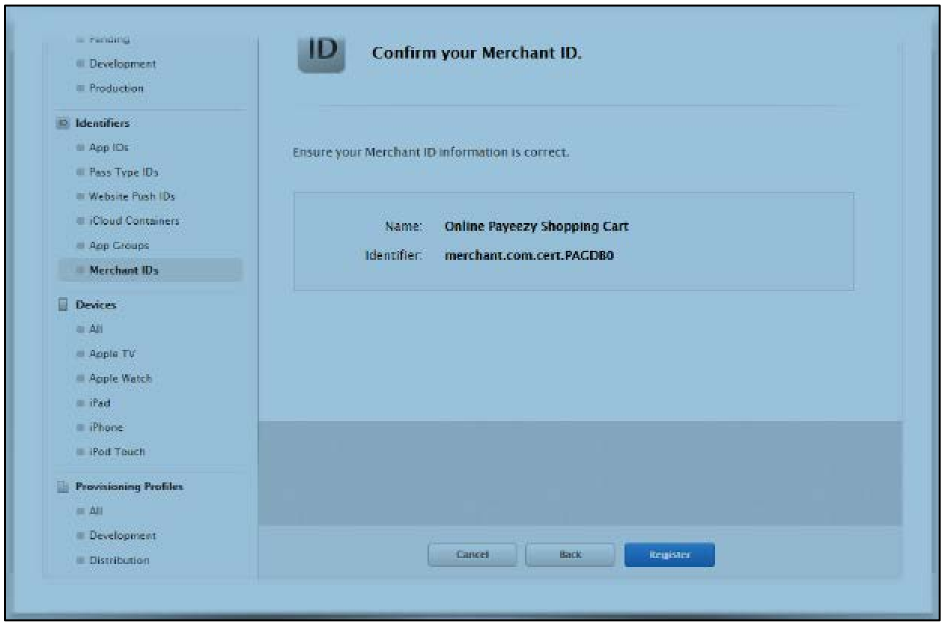
Merchant ID Description

Description: Online Payeezy Shopping Cart
You cannot use special characters such as @, &, *, ', "

Identifier
Enter a unique identifier for your Merchant ID, starting with the string 'merchant'.

ID: merchant.com.cert.PAGDB0
We recommend using a reverse-domain name style string (i.e., merchant.com.example.merchantname).

5. Click Continue, and then click Create Certificate. You may need the CSR from the First Data Developer Portal to create the Merchant Certificate.



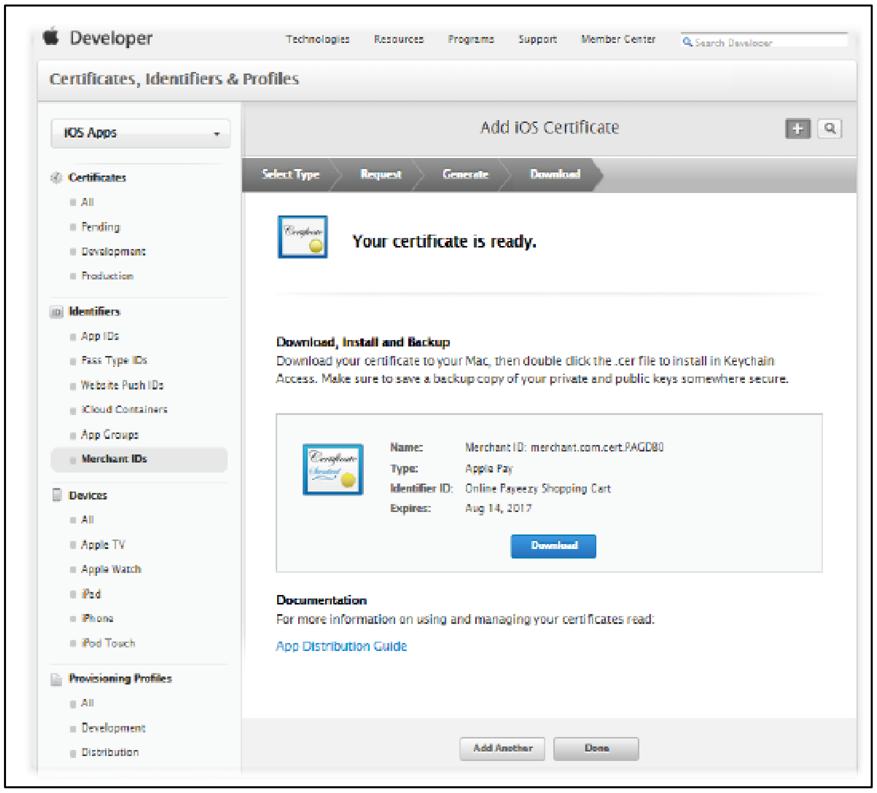
Confirm your Merchant ID.

Ensure your Merchant ID information is correct.

Name: Online Payeezy Shopping Cart
Identifier: merchant.com.cert.PAGDB0

6. Add Apple Pay Entitlement to your merchant certificate along with In-App Payment.

7. Download and install the certificate on your Mac.



Your certificate is ready.

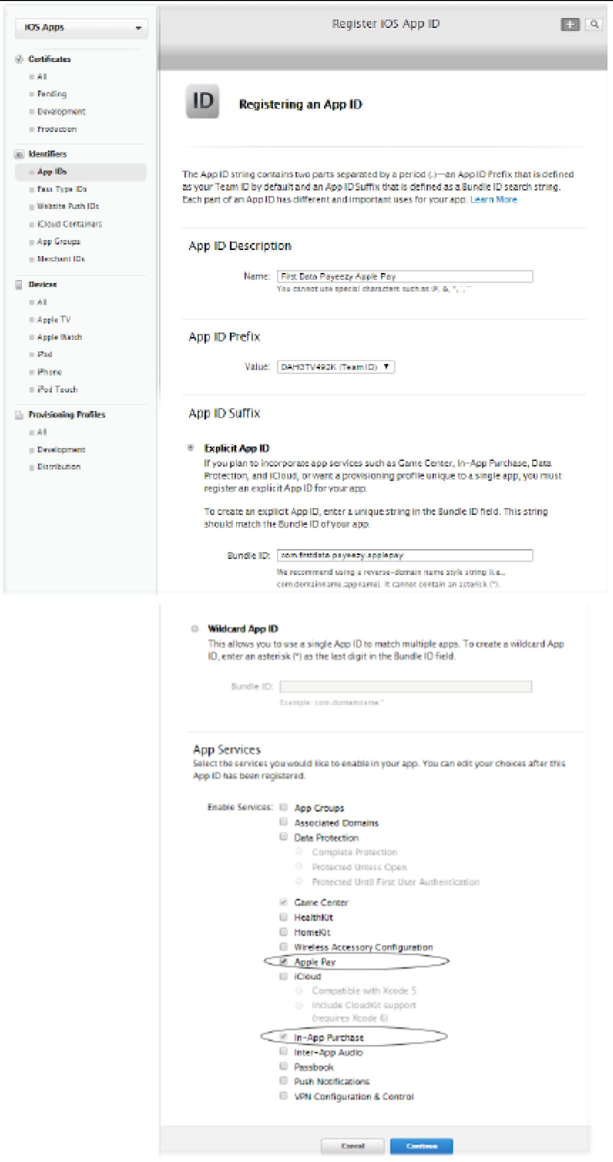
Download, Install and Backup
Download your certificate to your Mac, then double click the .cer file to install in Keychain Access. Make sure to save a backup copy of your private and public keys somewhere secure.

Name:	Merchant ID: merchant.com.cert.PAGD80
Type:	Apple Pay
Identifier ID:	Online Payeezy Shopping Cart
Expires:	Aug 14, 2017

Documentation
For more information on using and managing your certificates read:
App Distribution Guide

Provision Profile

The following figure shows your app's Bundle ID. You need to enable Apple Pay service for your app, and associate the Merchant to the app.



Registering an App ID

The App ID string contains two parts separated by a period (.)—an App ID Prefix that is defined as your Team ID by default and an App ID Suffix that is defined as a Bundle ID search string. Each part of an App ID has different and important uses for your app. Learn More

App ID Description

Name: First Data Payeezy Apple Pay

You cannot use special characters such as @, &, *, ', "

App ID Prefix

Value: DAH3TV492K (Team ID)

App ID Suffix

Explicit App ID

If you plan to incorporate app services such as Game Center, In-App Purchase, Data Protection, and iCloud, or want a provisioning profile unique to a single app, you must register an explicit App ID for your app.

To create an explicit App ID, enter a unique string in the Bundle ID field. This string should match the Bundle ID of your app.

Bundle ID: com.firstdata.payeezy.applepay

We recommend using a reverse-domain name style string (i.e., com.domainname.appname). It cannot contain an asterisk (*).

Wildcard App ID

This allows you to use a single App ID to match multiple apps. To create a wildcard App ID, enter an asterisk (*) as the last digit in the Bundle ID field.

Bundle ID:

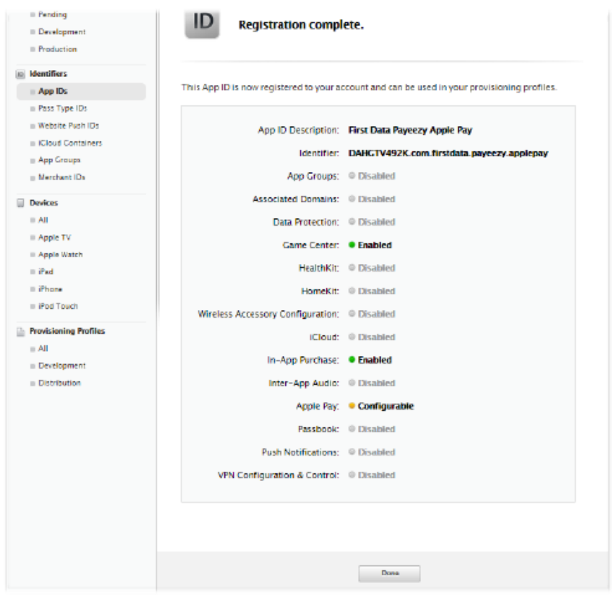
Example: com.domainname.*

App Services

Select the services you would like to enable in your app. You can edit your choices after this App ID has been registered.

Enable Services:
- App Groups
- Associated Domains
- Data Protection
  - Complete Protection
  - Protected Unless Open
  - Protected Until First User Authentication
- ✓ Game Center
- HealthKit
- HomeKit
- Wireless Accessory Configuration
- ✓ Apple Pay
- iCloud
  - Compatible with Xcode 5
  - Include CloudKit support (requires Xcode 6)
- ✓ In-App Purchase
- Inter-App Audio
- Passbook
- Push Notifications
- VPN Configuration & Control

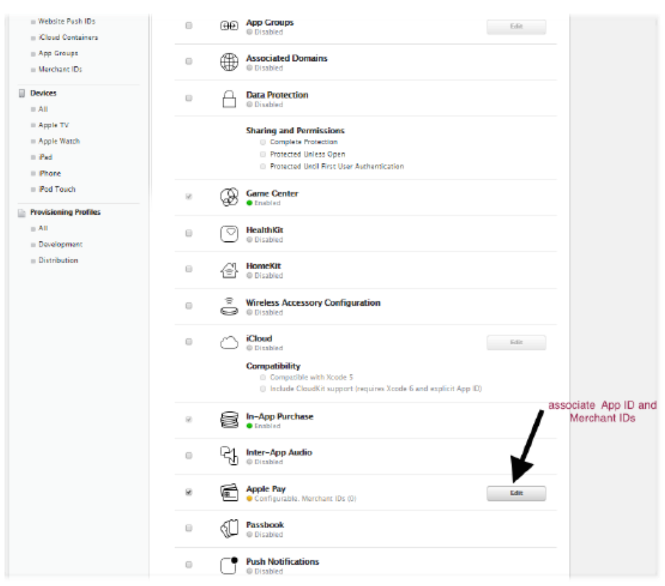
Associate App ID and Merchant IDs.



Registration complete.

This App ID is now registered to your account and can be used in your provisioning profiles.

App ID Description:	First Data Payeezy Apple Pay
Identifier:	DAHGTV492K.com.firstdata.payeezy.applepay
App Groups:	Disabled
Associated Domains:	Disabled
Data Protection:	Disabled
Game Center:	Enabled
HealthKit:	Disabled
HomeKit:	Disabled
Wireless Accessory Configuration:	Disabled
iCloud:	Disabled
In-App Purchase:	Enabled
Inter-App Audio:	Disabled
Apple Pay:	Configurable
Passbook:	Disabled
Push Notifications:	Disabled
VPN Configuration & Control:	Disabled



App Groups	Disabled
Associated Domains	Disabled
Data Protection	Disabled
Sharing and Permissions	Complete Protection; Protected Unless Open; Protected Until First User Authentication
✓ Game Center	Enabled
HealthKit	Disabled
HomeKit	Disabled
Wireless Accessory Configuration	Disabled
iCloud	Disabled
Compatibility	Compatible with Xcode 5; Include CloudKit support (requires Xcode 6 and explicit App ID)
✓ In-App Purchase	Enabled
Inter-App Audio	Disabled
✓ Apple Pay	Configurable, Merchant IDs (0)
Passbook	Disabled
Push Notifications	Disabled

Download and install your provision Profile. If you reopen your iOS project it should compile now.

Running the Apple Pay Sample App

After your project successfully compiles, you can run the sample app.

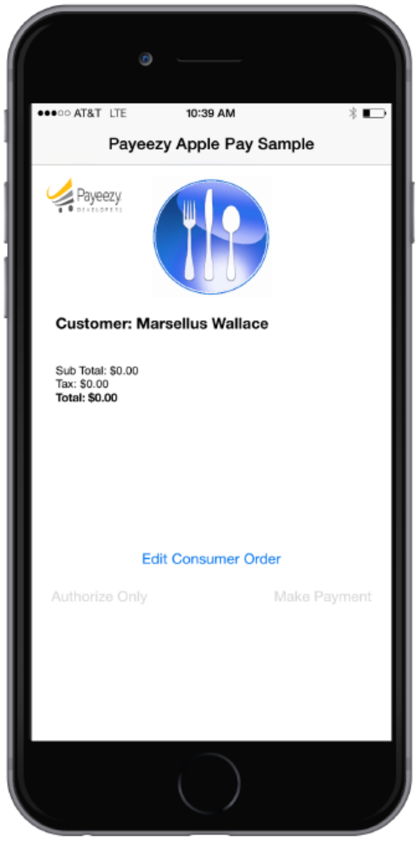
Note: The Apple Pay sample application is not compatible with simulators and may not run in one. You will need a compatible device to run the Apple Pay sample app. For details of acceptable devices, refer to Apple Pay (https://www.apple.com/apple-pay/) and search for compatible devices.

To run the sample app:

1. Add a credit card to your passbook and authorize the card for Apple Pay.
2. Connect your iPhone 6/6+ and run the app.

Refer to MerchantMakePaymentViewController.m for code implementation details
FDInAppPaymentProcessor()

For more details on the code and sample, refer to the Apple Pay Quick Start Guide (https://github.com/payeezy/payeezy_apple_pay/blob/master/guide/apple_pay_quick_start(2.0).pdf)

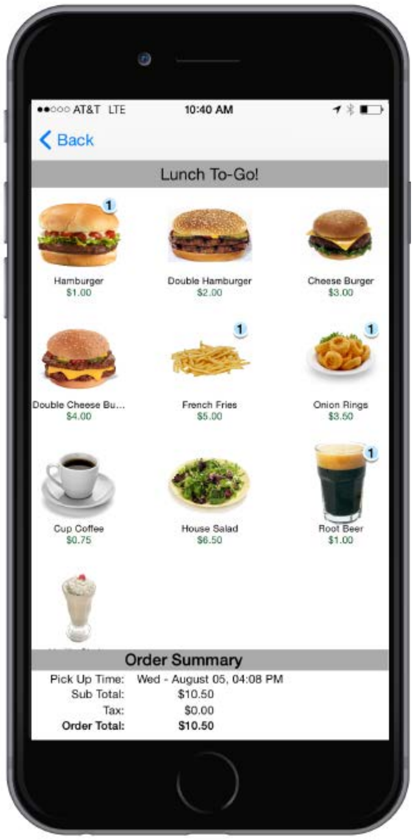


●●●○○ AT&T LTE 10:39 AM

Payeezy Apple Pay Sample

Customer: Marsellus Wallace

Sub Total: $0.00
Tax: $0.00
Total: $0.00

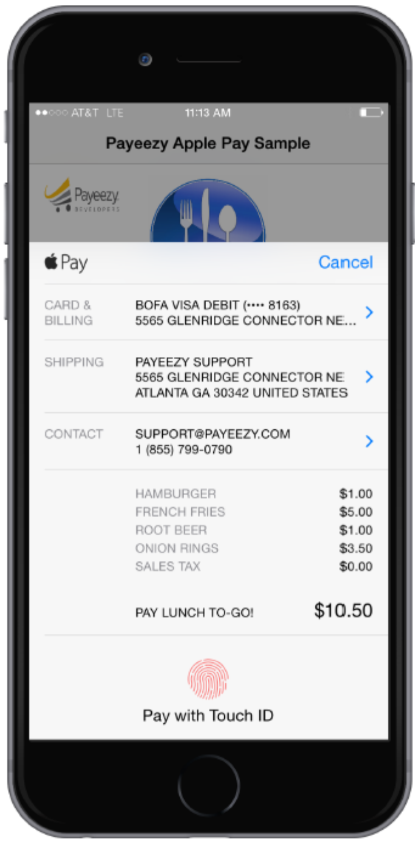


●●○○○ AT&T LTE 10:40 AM

Hamburger $1.00	Double Hamburger $2.00	Cheese Burger $3.00
Double Cheese Bu... $4.00	French Fries $5.00	Onion Rings $3.50
Cup Coffee $0.75	House Salad $6.50	Root Beer $1.00

Order Summary

Pick Up Time: Wed - August 05, 04:08 PM
Sub Total: $10.50
Tax: $0.00
Order Total: $10.50



●●○○○ AT&T LTE 11:13 AM

Payeezy Apple Pay Sample

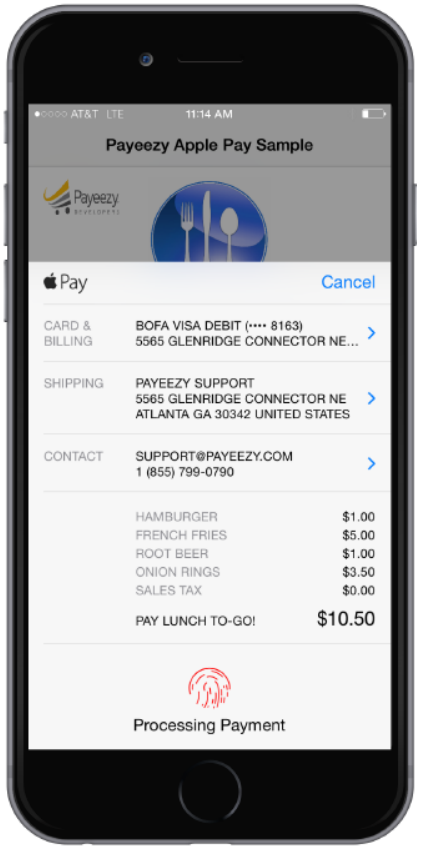
CARD & BILLING: BOFA VISA DEBIT (•••• 8163) 5565 GLENRIDGE CONNECTOR NE...

SHIPPING: PAYEEZY SUPPORT 5565 GLENRIDGE CONNECTOR NE ATLANTA GA 30342 UNITED STATES

CONTACT: SUPPORT@PAYEEZY.COM 1 (855) 799-0790

HAMBURGER	$1.00
FRENCH FRIES	$5.00
ROOT BEER	$1.00
ONION RINGS	$3.50
SALES TAX	$0.00
PAY LUNCH TO-GO!	$10.50

Pay with Touch ID



●○○○○ AT&T LTE 11:14 AM

Payeezy Apple Pay Sample

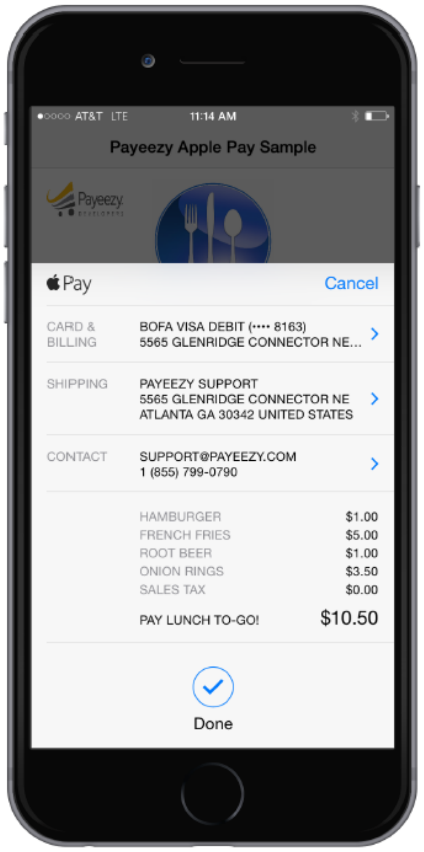
CARD & BILLING: BOFA VISA DEBIT (•••• 8163) 5565 GLENRIDGE CONNECTOR NE...

SHIPPING: PAYEEZY SUPPORT 5565 GLENRIDGE CONNECTOR NE ATLANTA GA 30342 UNITED STATES

CONTACT: SUPPORT@PAYEEZY.COM 1 (855) 799-0790

HAMBURGER	$1.00
FRENCH FRIES	$5.00
ROOT BEER	$1.00
ONION RINGS	$3.50
SALES TAX	$0.00
PAY LUNCH TO-GO!	$10.50

Processing Payment



●○○○○ AT&T LTE 11:14 AM

Payeezy Apple Pay Sample

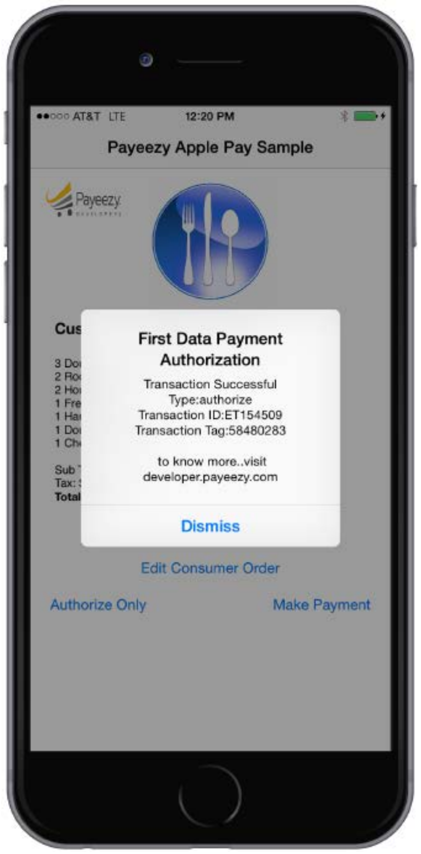
CARD & BILLING: BOFA VISA DEBIT (•••• 8163) 5565 GLENRIDGE CONNECTOR NE...

SHIPPING: PAYEEZY SUPPORT 5565 GLENRIDGE CONNECTOR NE ATLANTA GA 30342 UNITED STATES

CONTACT: SUPPORT@PAYEEZY.COM 1 (855) 799-0790

HAMBURGER	$1.00
FRENCH FRIES	$5.00
ROOT BEER	$1.00
ONION RINGS	$3.50
SALES TAX	$0.00
PAY LUNCH TO-GO!	$10.50

Done



●●○○○ AT&T LTE 12:20 PM

Payeezy Apple Pay Sample

First Data Payment Authorization

Transaction Successful
Type:authorize
Transaction ID:ET154509
Transaction Tag:58480283

to know more..visit developer.payeezy.com

On the Web Integration

Apple lists three requirements for using Apple Pay on a website:

- The developer must have an Apple Developer Account
- All pages must be served over HTTPS
- Your website must comply with Apple Pay guidelines.

Refer to the Apple Pay JS Guide for more information.

Apple Certificates, Identifiers and Profiles

Once you have downloaded a CSR at the First Data Developer Portal, log in to Apple's Developer Portal and use it to generate a Merchant certificate and a Payment Processing certificate. These steps are the same as the ones for an In-App integration.

For an On the Web integration, you will also need to register and verify the domains that will be processing transactions and create a Merchant Identity certificate.

Register Merchant ID

To register a Merchant ID:

1. Sign in to developer.apple.com, or register for a developer account if you do not have one.
2. In the Member Center, click **Certificates, Identifiers & Profiles**.
3. Click **Identifiers**.
4. Click **Merchant IDs**, and then click the **Add** button in the upper right corner.
5. Enter the Merchant Description/Name and Identifier.
   Note: The Merchant ID must match the App Lable from the My Merchants page in the First Data Developer Portal.
6. Click **Register**, and then click **Done.**

For more information, refer to the Apple Pay Programming Guide.

Create Payment Processing Certificate

To create a Payment Processing certificate:

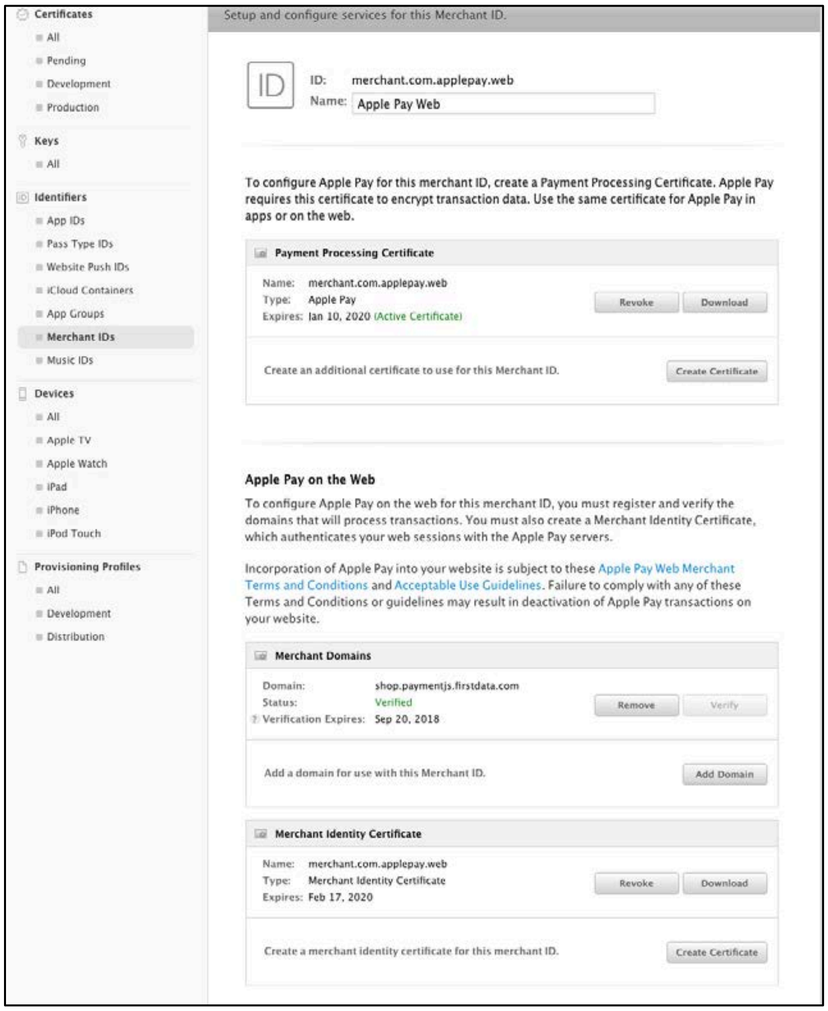
1. Select the Merchant ID you have just registered, and click **Edit**.
2. In the Payment Processing Certificates section, click **Choose File**, select the CSR you downloaded from the First Data Developer Portal, and click Generate.
3. Click **Download**, and then **Done**.

For more information, refer to the Apple Pay Programming Guide.

Register and Verify Domains and Creating Merchant Identity Certificate

To register and verify a domain and create a Merchant Identity certificate:

1. Select the Merchant ID you have just registered, and click **Edit**.
2. In the Apple Pay on the Web section, click **Add Domain**.
3. Enter the domain name and click **Continue**.
4. Download the file created by the site and host it at the location you entered.
5. Once the file is available on your server, click **Verify**. If the verification succeeds, the Merchant ID Settings page shows a green Verified label.
6. In the Apple Pay on the Web section, click **Create Certificate**.
7. Follow the instructions to download the Merchant Identity Certificate.



Setup and configure services for this Merchant ID.

ID: merchant.com.applepay.web
Name: Apple Pay Web

To configure Apple Pay for this merchant ID, create a Payment Processing Certificate. Apple Pay requires this certificate to encrypt transaction data. Use the same certificate for Apple Pay in apps or on the web.

Payment Processing Certificate

Name: merchant.com.applepay.web
Type: Apple Pay
Expires: Jan 10, 2020 (Active Certificate)

Create an additional certificate to use for this Merchant ID.

Apple Pay on the Web

To configure Apple Pay on the web for this merchant ID, you must register and verify the domains that will process transactions. You must also create a Merchant Identity Certificate, which authenticates your web sessions with the Apple Pay servers.

Incorporation of Apple Pay into your website is subject to these Apple Pay Web Merchant Terms and Conditions and Acceptable Use Guidelines. Failure to comply with any of these Terms and Conditions or guidelines may result in deactivation of Apple Pay transactions on your website.

Merchant Domains

Domain: shop.paymentjs.firstdata.com
Status: Verified
Verification Expires: Sep 20, 2018

Add a domain for use with this Merchant ID.

Merchant Identity Certificate

Name: merchant.com.applepay.web
Type: Merchant Identity Certificate
Expires: Feb 17, 2020

Create a merchant identity certificate for this merchant ID.

For more information, refer to the Apple Pay JS Guide.

Apple Pay On the Web and Sample Application Files

The files shown are a part of a sample PHP-based application that has integrated Apple Pay On the Web using the Laravel framework.

Note that Apple Pay On the Web requires the Safari browser and valid payment cards associated with the account before the Apple Pay button is displayed. To make this happen for a sample app:

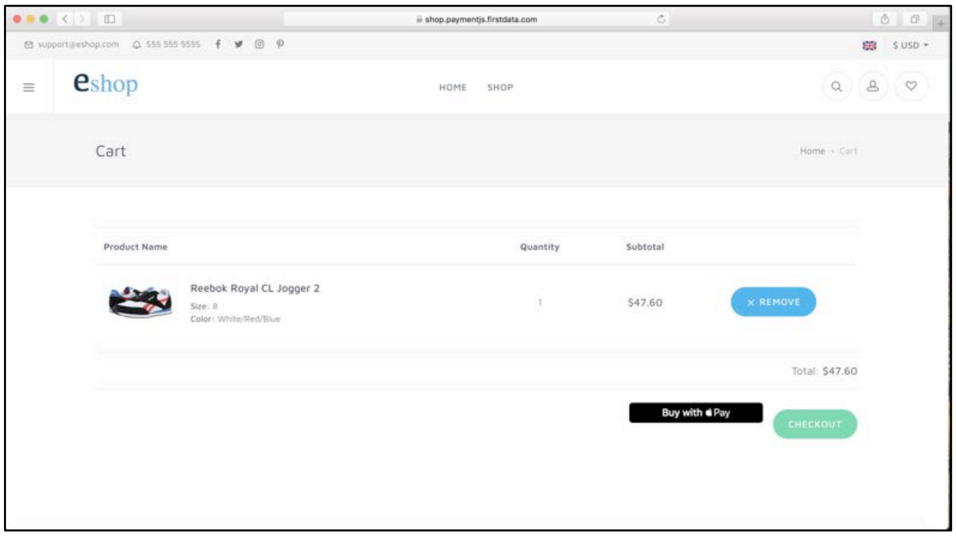
1. Create a sandbox test user account in iTunes.
2. Use the sandbox test user account to log into iCloud on your devices (for example, your Mac and your iPhone.)

In the included sample application files, refer to the file `index.js` for the code necessary to:

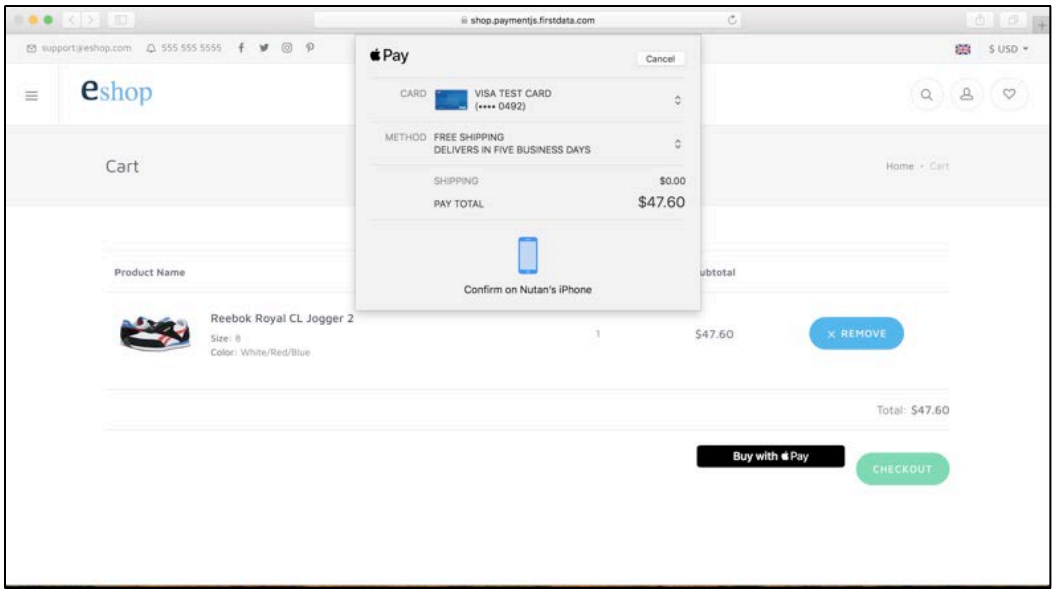
- Display the Apple Pay button
- Initiate the payment request when the consumer clicks the button
- Validate the merchant
- Process the request through First Data's payment processor.

In the included files:

1. In the `index.js` file, ensure that valid test cards have been added to the wallet associated with the sandbox test user account. (Review `canMakePaymentsWithActiveCard`.) This allows the Apple Pay button to be displayed on the screen.

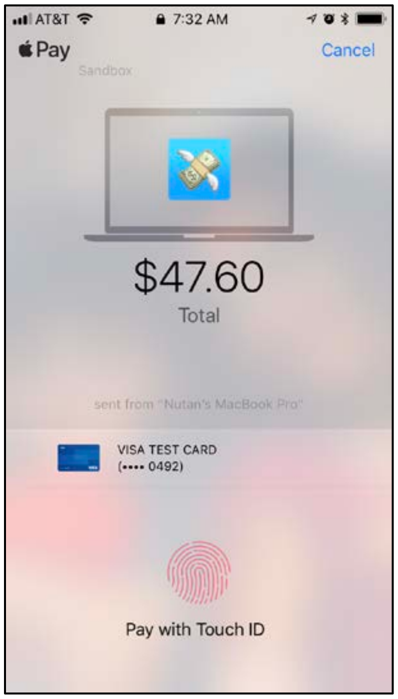


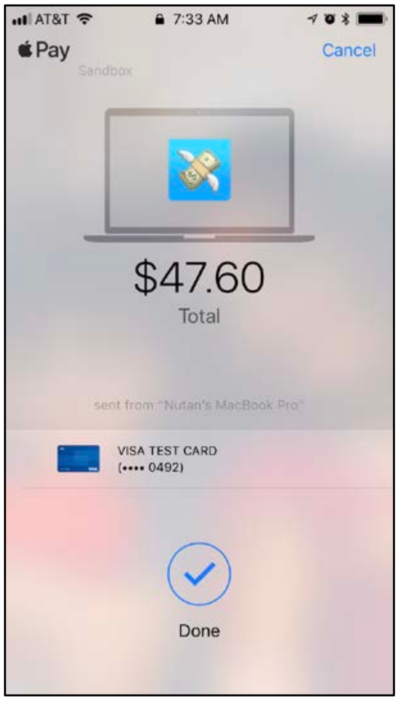
2. When the consumer clicks the Apple Pay button, the function `applePayButtonClicked` is called. This creates a new payment request and Apple Pay session and displays the payment sheet.



3. The merchant validation is carried out at the server side. The sample app makes a back-end call using `getApplePaySession.` This is a POST endpoint to obtain a merchant session for Apple Pay. The client provides the URL to call in its body. The app must provide Merchant ID, Apple Pay Certificate, Domain Name and Display Name as data. These values must match the data provided during Apple configuration, above. Apple returns:
   - Nonce
   - Merchant session identifier
   - Signature.
   - Ephemeral public key
   - Encrypted payload.

   For details, refer to the Apple Pay API and sample payload on the First Data Developer Portal.
4. The next step is Payment Authorization. The response from Apple must be mapped to the First Data Apple Pay authorization request requirements, as described on the First Data Developer Portal. This request is submitted to the appropriate First Data endpoint.
5. First Data processes the request and returns a success or failure message, as appropriate. Once the authorization is successful, the consumer completes the transaction on their device and receives the confirmation screen, also on their device. See the screen captures below.


AT&T
7:32 AM
Pay
Cancel
Sandbox
$47.60
Total
sent from "Nutan's MacBook Pro"
VISA TEST CARD
(•••• 0492)
Pay with Touch ID


AT&T
7:33 AM
Pay
Cancel
Sandbox
$47.60
Total
sent from "Nutan's MacBook Pro"
VISA TEST CARD
(•••• 0492)
Done

Example Files

The files shown here are part of an example PHP-based application. They pertain to the integration of Apple Pay On the Web.

index.js

```
/*
        Copyright (C) 2016 Apple Inc. All Rights Reserved.
        See LICENSE.txt for this sample’s licensing information

        Abstract:
        The main client-side JS. Handles displaying the Apple Pay button and requesting a payment.
        */
        /**
        * This method is called when the page is loaded.
        * We use it to show the Apple Pay button as appropriate.
        * Here we're using the ApplePaySession.canMakePayments() method,
        * which performs a basic hardware check.
        *
        * If we wanted more fine-grained control, we could use
        * ApplePaySession.canMakePaymentsWithActiveCards() instead.
        */
        /*document.addEventListener('DOMContentLoaded', () => {
                if (window.ApplePaySession) {
                        if (ApplePaySession.canMakePayments) {
                                showApplePayButton();
                        }
                }
        });
        */
        document.addEventListener('DOMContentLoaded', () => {
                if (window.ApplePaySession) {
                        //console.log("canMakePaymentsWithActiveCard");
                        var merchantIdentifier = 'merchant.com.applepay.web';
                        var promise = ApplePaySession.canMakePaymentsWithActiveCard(merchantIdentifier);
                        //console.log(promise);
                        promise.then(function (canMakePayments) {
                                //console.log(canMakePayments);
                                        if (canMakePayments)
                                        showApplePayButton();
                }); }
        });
        function showApplePayButton() {
                HTMLCollection.prototype[Symbol.iterator] = Array.prototype[Symbol.iterator];
                var buttons = document.getElementsByClassName("apple-pay-button");
```

```
		for (let button of buttons) {
			button.className += " visible";
		}
	}
	/**
	* Apple Pay Logic
	* Our entry point for Apple Pay interactions.
	* Triggered when the Apple Pay button is pressed
	*/
	function applePayButtonClicked() {
		var paymentRequest = {
			countryCode: 'US',
			currencyCode: 'USD',
			shippingMethods: [
			{
				label: 'Free Shipping',
				amount: '0.00',
				identifier: 'free',
				detail: 'Delivers in five business days'
				},
			{
				label: 'Express Shipping',
				amount: '5.00',
				identifier: 'express',
				detail: 'Delivers in two business days'
				},
			],

			lineItems: [
			{
				label: 'Shipping',
				amount: '0.00'
				}
			],
			total: {
				label: 'Total',
				type: 'final',
				amount: '47.60'
			},
			supportedNetworks:[ 'amex', 'discover', 'masterCard', 'visa'],
			merchantCapabilities: [ 'supports3DS' ]
		};
		//console.log("supports version: " + ApplePaySession.supportsVersion(2));
```

```
        var session = new ApplePaySession(2, paymentRequest);

        session.oncancel = function(event){
                console.log("oncancel event triggered");
                console.log(event);
        }
        /**
        * Merchant Validation
        * We call our merchant session endpoint, passing the URL to use
        */
        session.onvalidatemerchant = function(event){
                //console.log("Validate merchant : ");
                var validationURL = event.validationURL;
                //console.log(validationURL);
                getApplePaySession(validationURL).then(function(response) {
                        //console.log(response);
                        session.completeMerchantValidation(response);
                });
        };

        /**
        * Shipping Method Selection
        * If the user changes their chosen shipping method we need to recalculate
        * the total price. We can use the shipping method identifier to determine
        * which method was selected.
        */
        session.onshippingmethodselected = (event) => {
                const shippingCost = event.shippingMethod.identifier === 'free' ? '0.00' : '5.00';
                const totalCost = event.shippingMethod.identifier === 'free' ? '47.60' : '52.60';
                const lineItems = [
                        {
                                label: 'Shipping',
                                amount: shippingCost,
                        },
                ];
                const total = {
                        label: 'Total',
                        amount: totalCost,
                };
                session.completeShippingMethodSelection(ApplePaySession.STATUS_SUCCESS, total,
lineItems);
        };
        /**
```

```
* Payment Authorization
* Here you receive the encrypted payment data. You would then send it
* on to your payment provider for processing, and return an appropriate
* status in session.completePayment()
*/
session.onpaymentauthorized = function(event){
        // Send payment for processing...
        //console.log(event);
        const payment = event.payment;
        //console.log('payment');
        //console.log(payment);
        processPaymentRequest(payment).then(function(response){
                //console.log(response);
                // ...return a status and redirect to a confirmation page
                if(response["validation_status"] === "success"){
                        session.completePayment(ApplePaySession.STATUS_SUCCESS);
                        //window.location.href = "../success.html";
                } else {
                        session.completePayment(ApplePaySession.STATUS_FAILURE);
                }
        });

}
// All our handlers are setup - start the Apple Pay payment
session.begin();
```

First Data

support.js

```
/*
        Copyright (C) 2016 Apple Inc. All Rights Reserved.
        See LICENSE.txt for this sample’s licensing information

        Abstract:
        A helper function that requests an Apple Pay merchant session using a promise.
        */
        function getApplePaySession(url) {
            console.log("getApplePaySession");
            return new Promise(function (resolve, reject) {
            var xhr = new XMLHttpRequest();
            token = document.querySelector('meta[name="csrf-token"]').content;
            xhr.open('POST', '/getApplePaySession');
            xhr.onload = function () {
              if (this.status >= 200 && this.status < 300) {
                //console.log("Returned response from the server: ");
               //console.log(xhr.response);
               resolve(JSON.parse(xhr.response));
              } else {
                reject({
                  status: this.status,
                  statusText: xhr.statusText
                });
              }
            };
            xhr.onerror = function () {
              reject({
                status: this.status,
                statusText: xhr.statusText
              });
            };
            xhr.setRequestHeader("Content-Type", "application/json");
            xhr.setRequestHeader('X-CSRF-TOKEN', token);
            xhr.send(JSON.stringify({url: url}));
          });
        }
        function processPaymentRequest(request) {
            //console.log("processPaymentRequest");
            return new Promise(function (resolve, reject) {
            var xhr = new XMLHttpRequest();
            token = document.querySelector('meta[name="csrf-token"]').content;
            xhr.open('POST', '/processPaymentRequest');
            xhr.onload = function () {
              if (this.status >= 200 && this.status < 300) {
                //console.log("Returned response from the server: ");
                //console.log(xhr.response);
                resolve(JSON.parse(xhr.response));
              } else {
                reject({
                  status: this.status,
                  statusText: xhr.statusText
                });
              }
            };
            xhr.onerror = function () {
              reject({
                status: this.status,
                statusText: xhr.statusText
              });
            };
            xhr.setRequestHeader("Content-Type", "application/json");
            xhr.setRequestHeader('X-CSRF-TOKEN', token);
            xhr.send(JSON.stringify({request: request}));
          });
        }
```