Apple Pay Essentials: Sample Code

Configure Your Development Environment

The Red project contains the files that make up the client/server sample code for *Apple Pay Essentials*. This project has been tested with OS X (10.10.5), Xcode 7.2.1, and iOS 9.0.2.

Perform the steps described in the following sections to configure the server (Red) and client (MerchantApp) apps.

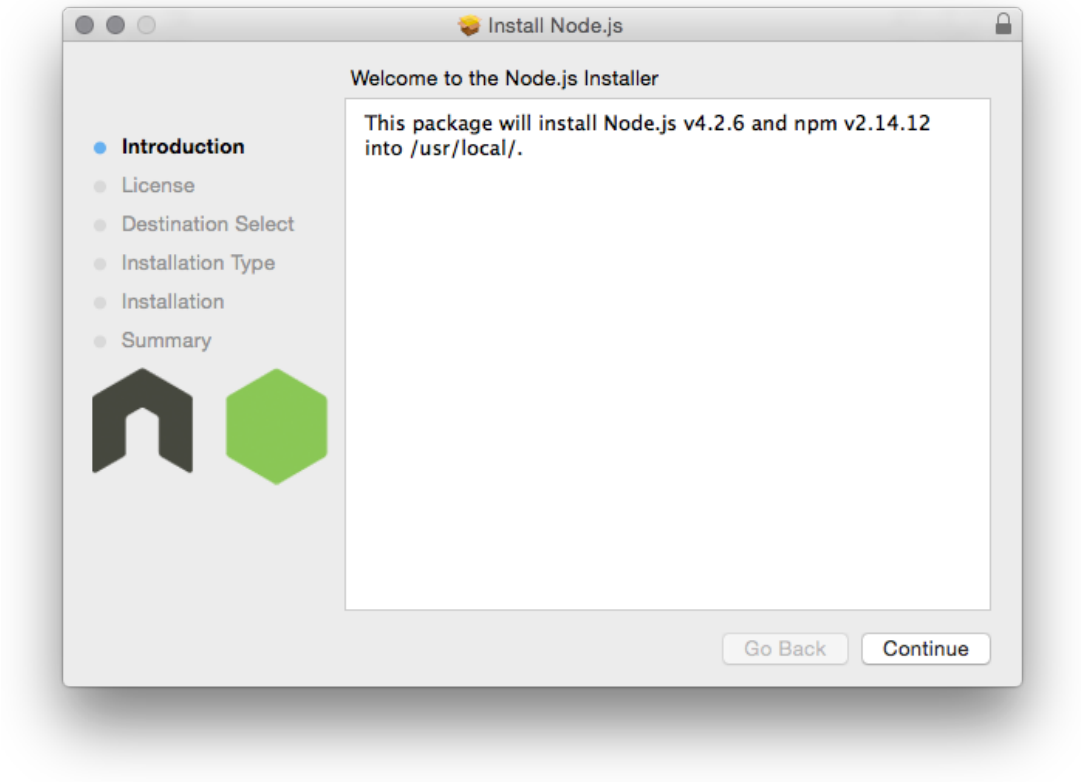
Note:
This sample code uses the Stripe payment gateway to perform Apple Pay transactions. To run the apps with minimal modifications to the source files, you should open a Stripe merchant account. If you have accounts with other payment gateways, follow the appropriate steps to get the required Apple Pay merchant certificates. You must also adapt the source code in the web app and the iOS app to use the appropriate payment gateway API.

Install Node.js

If you do not have Node installed on your development Mac, you must install it for the sample code to work.

To install Node:

1. Download the installation package from https://node.org.
2. Open the installation package, and follow the instructions.



Install MongoDB

MongoDB is the document-based database used as the datastore for the webapp that serves the client iOS app.

To install MongoDB:

1. Download the MongoDB binaries from https://mongodb.org.
2. Open (decompress) the TGZ file.
3. In a Terminal window or tab, copy the binaries to the `/usr/local/bin` directory:

```
$ cd <download_location>/mongodb-osx-x86_64-3.2.1/bin # or the appropriate directory
$ sudo cp * /usr/local/bin                            # requires an administrator password
```

Add Your Mac's IP Address to the /etc/hosts File

The MerchantApp app uses the `http://red:12345` URI to communicate with the server webapp (`red`) running in your development Mac. For the app to locate the `red` process using the aforementioned URI, the `/etc/hosts` file in your Mac must have a line similar to this (where `<target_address>` is your Mac's IP address):

```
##
# Host Database
#
# localhost is used to configure the loopback interface
# when the system is booting.  Do not change this entry.
##
127.0.0.1            localhost
255.255.255.255      broadcasthost
::1                  localhost
<target_address>     red
```

To determine your Mac's IP address, go to System Preferences > Network, and select the first, active network service. If you have more than one active service, such as a wired service and a wireless service, you can use any of them (for wireless services, click the Advanced button and then TCP/IP to get the IP address.) If you can, you should use a fixed address that does not change between reboots.

For example, if your local IP address is 100.200.10.5, you can add that IP address and the `red` hostname to the hosts file by entering these commands in a Terminal window:

```
[ernest@Brook:/]$ cd /etc
[ernest@Brook:/etc]$ su
Password:                                  # enter the root user password (see the next section)
[root@Brook:/etc]$ echo '100.200.10.5       red' >> hosts
[root@Brook:/etc]$ cat hosts
##
# Host Database
#
# localhost is used to configure the loopback interface
# when the system is booting.  Do not change this entry.
##
127.0.0.1          localhost
255.255.255.255    broadcasthost
::1                localhost
100.200.10.5       red
[root@Brook:/etc]$ exit
[ernest@Brook:/etc]$
```

Enable root User

The `su` command requires that the root user be enabled. If you do not have the root user enabled in the proxy-server Mac, you can enable it temporarily.

To enable the root user, enter this command in a Terminal window:

```
$ dsenableroot
username = ernest
user password:                          # enter your user password
root password:                          # enter a password for the root user
verify root password:                   # re-enter the password for the root user

dsenableroot:: ***Successfully enabled root user.
```

After you are done modifying the hosts file, you should disable the root user.

To disable the root user, enter this command in a Terminal window:

```
[ernest@Brook:/etc]$ dsenableroot -d
username = ernest
user password:                          # enter your user password

dsenableroot:: ***Successfully disabled root user.
```

Set the Stripe Key in the Server Web App

The stripe variable in the `<red_project_dir>/server_app/red.js` file needs to be set to your test Stripe key. Replace `<my_key>` with your key's string.

Update Node Modules

To update the Node modules that the webapp requires, in a Terminal window, execute these commands:

```
$ cd <red_project_dir>/server_app
$ npm update
```

Start Mongo Daemon

The Mongo daemon (`mongod`) runs the Mongo database engine. To start the Mongo daemon, in a Terminal window execute these commands:

```
$ cd <red_project_dir>
$ mongod --dbpath data --port 27017
```

Start the Red Webapp

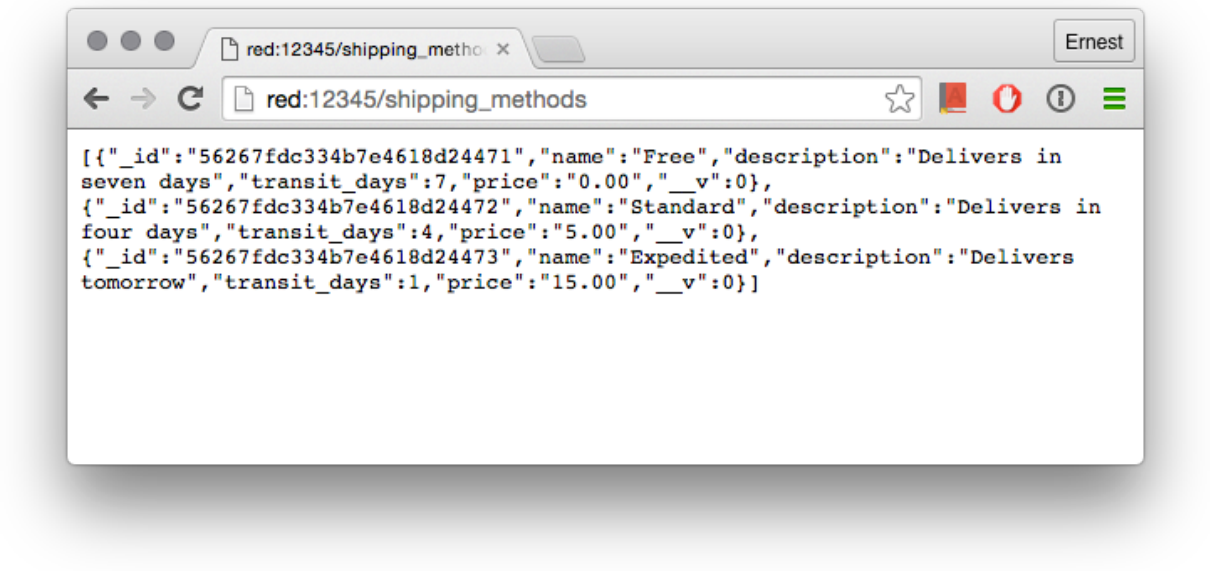
The `red.js` script is the webapp to which the client iOS app connects to get product data and submit order and payment information.

After starting the Mongo daemon, start the red webapp. The first time it runs (the `<red_project_dir>/data` and `<red_project_dir>/data/db` directories are empty), the webapp initializes the collections that contain product, and shipping-method details.

To start the webapp:

```
$ cd <red_project_dir>/server_app
$ node red.js
http://red:12345
initializing product collection
initializing shipping-method collection
^C                                      # stop the app after it initializes the collections
$ node red.js                           # restart it
http://red:12345
```

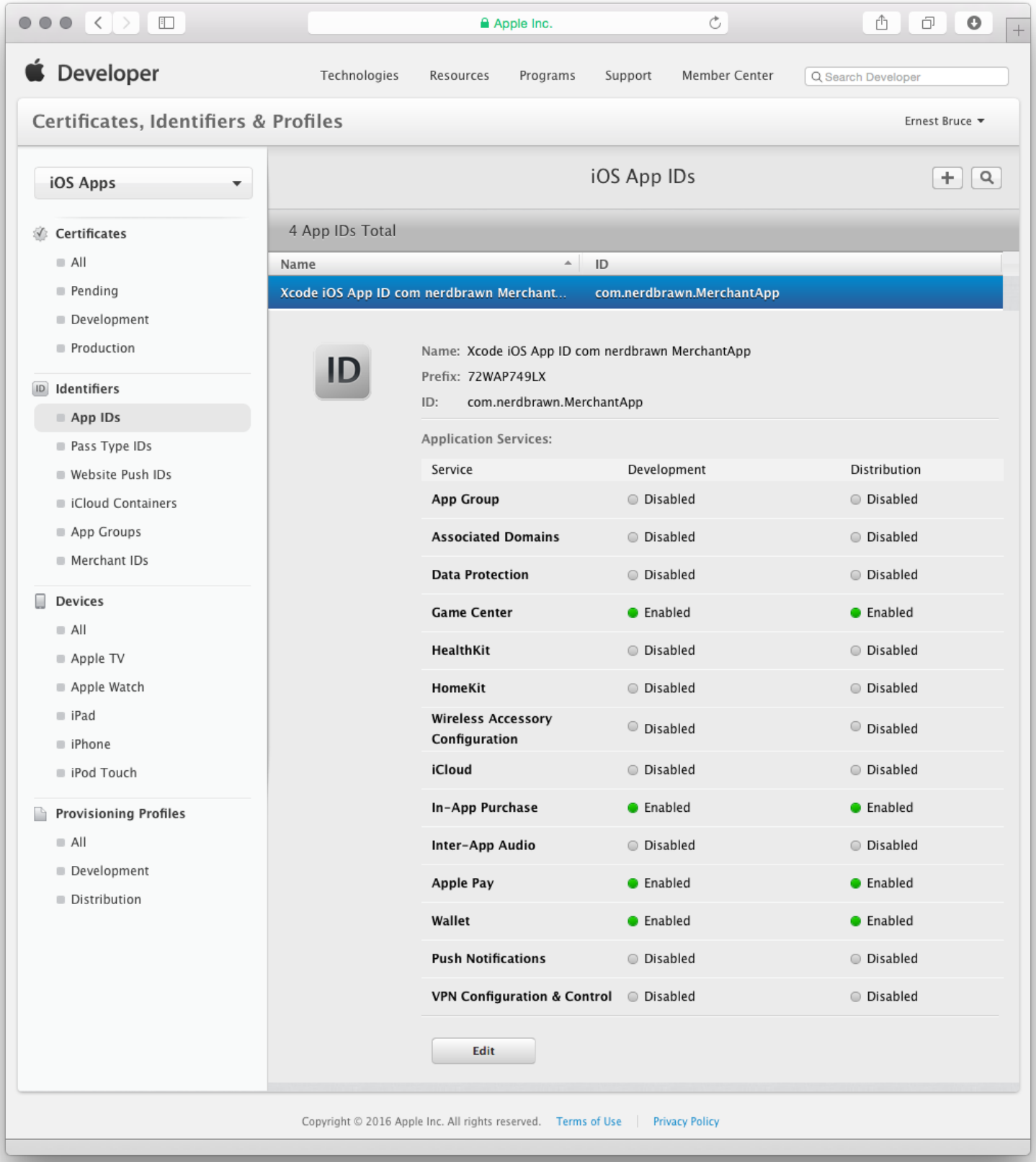
To ensure that the webapp is running correctly, enter this URI in a web browser: http://red:12345/shipping_methods. The result should be similar to this:



Set the App ID for the MerchantApp App in Member Center

All app projects must be configured with an appropriate bundle identifier (App ID).

In Member Center <https://developer.apple.com/membercenter>, ensure that there is an App ID for MerchantApp, such as `com.<your_company>.MerchantApp` for your team.


Apple Inc.
Developer
Technologies
Resources
Programs
Support
Member Center
Search Developer
Certificates, Identifiers & Profiles
Ernest Bruce
iOS Apps
Certificates
All
Pending
Development
Production
Identifiers
App IDs
Pass Type IDs
Website Push IDs
iCloud Containers
App Groups
Merchant IDs
Devices
All
Apple TV
Apple Watch
iPad
iPhone
iPod Touch
Provisioning Profiles
All
Development
Distribution
iOS App IDs
4 App IDs Total
Name
ID
Xcode iOS App ID com nerdbrawn Merchant...
com.nerdbrawn.MerchantApp
ID
Name: Xcode iOS App ID com nerdbrawn MerchantApp
Prefix: 72WAP749LX
ID: com.nerdbrawn.MerchantApp
Application Services:
Service
Development
Distribution
App Group
Disabled
Disabled
Associated Domains
Disabled
Disabled
Data Protection
Disabled
Disabled
Game Center
Enabled
Enabled
HealthKit
Disabled
Disabled
HomeKit
Disabled
Disabled
Wireless Accessory Configuration
Disabled
Disabled
iCloud
Disabled
Disabled
In-App Purchase
Enabled
Enabled
Inter-App Audio
Disabled
Disabled
Apple Pay
Enabled
Enabled
Wallet
Enabled
Enabled
Push Notifications
Disabled
Disabled
VPN Configuration & Control
Disabled
Disabled
Edit
Copyright © 2016 Apple Inc. All rights reserved.
Terms of Use
Privacy Policy

Configure Your Development Mac as a Proxy Server

To run the MerchantApp app on an iOS device, you need a *proxy server*. A proxy server can redirect certain HTTP requests that would go to a wide network, such as the internet, to a particular computer. In this case, HTTP requests from MerchantApp running on your development iOS device should go to your development computer, where the server webapp runs. You can configure your development Mac as a proxy server that that intercepts requests form your iOS device and forwards them to the network, except requests targeted to the `red` hostname. This section describes how to set up your development Mac as a proxy server, and how to set up your iOS device so that it uses the proxy server.

Get Your iOS Device's IP Address

To get the IP address of your development iOS device:

1. Open Settings > Wi-Fi.
2. Tap the info button next to the wireless networks to which the iOS is connected.
3. Copy the IP address in the IP Address field.

Configure SquidMan on Your Development Mac

SquidMan is an app that installs and manages the Squid proxy cache.

To configure SquidMan on your development Mac:

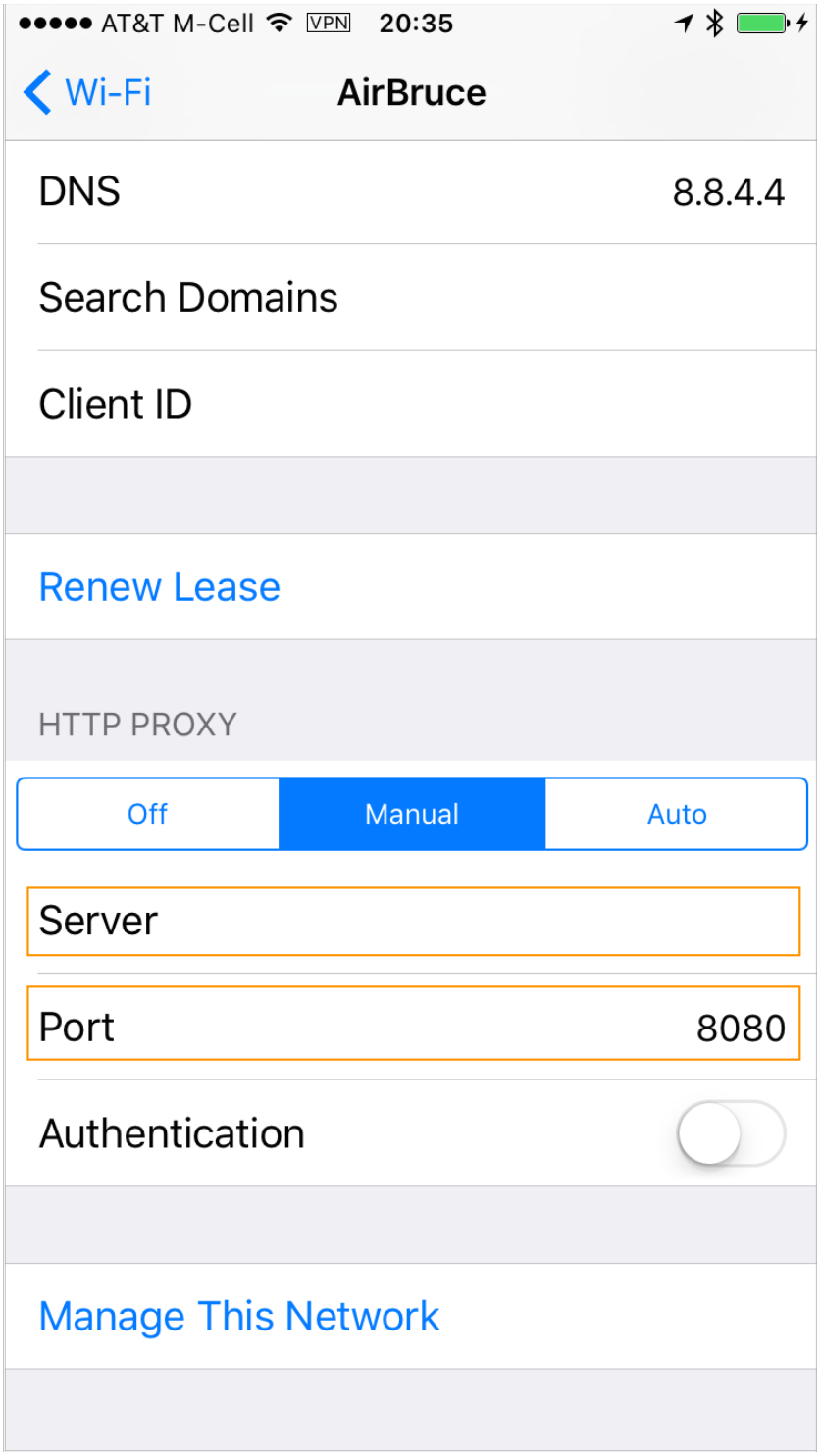
1. Download the SquidMan proxy-server software from http://squidman.net/squidman.
2. Copy the SquidMan app to your `Applications` directory.
3. Open SquidMan, choose SquidMan > Preferences, and click Clients.
4. Add your development iOS device IP address to the "Provide proxy services for" list. (You can also specify a subnet, as the app examples describe.)
5. Close the Preferences window.
6. In the SquidMan window, click Start Squid.

Configure Your iOS Device to Use Your Proxy Server

To configure your development iOS device use the proxy server:

1. Open Settings > Wi-Fi.
2. Tap the info button next to the wireless networks to which the iOS is connected.
3. Scroll to the HTTP PROXY section at bottom of the screen, and tap Manual.
4. In the Server field, enter the IP address of your development Mac.
5. In the Port field, enter `8080`.

6. Make sure that Authentication is turned off.



7. Tap the Wi-Fi back button.

Configure the MerchantApp Project

The following sections describe how to configure the MerchantApp project so that you can run it on an iOS simulator or device.

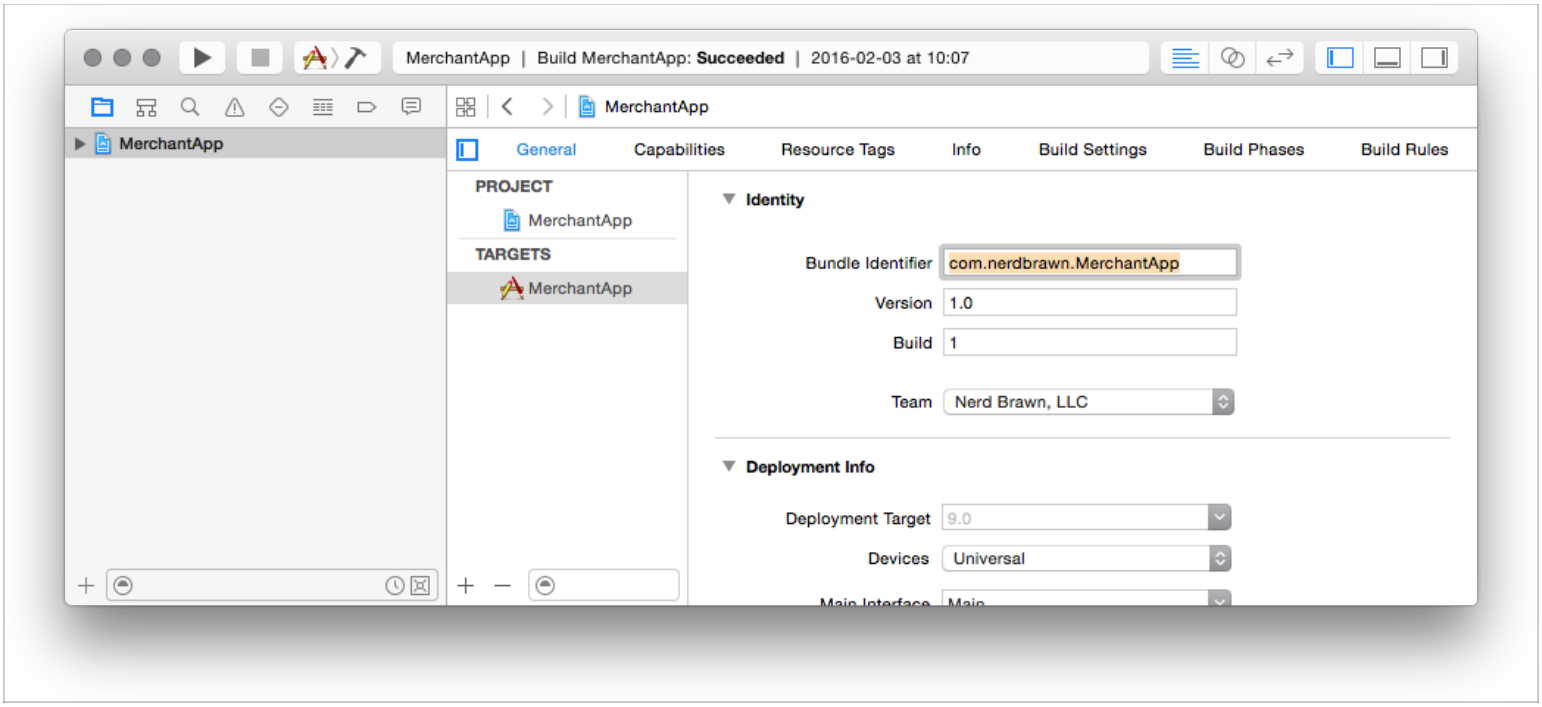
Important:
Your development Apple ID must be present in your Xcode account list so that Xcode can access your Developer Program resources. Ensure that your development Apple ID is listed in your accounts list in Preferences > Accounts.

Set the Project's Bundle Identifier to the MerchantApp App ID

You need to set the MerchantApp project'a bundle identifier, which looks like `com.<my_company>.MerchantApp`, to the MerchantApp's App ID, which you set earlier in the "Set the App ID for the MerchantApp App in Member Center" section.

To set the bundle identifier in the MerchantApp project:

1. Open the MerchantApp project in `<red_project_dir>/client_app/MerchantApp.xcodeproj`.
2. In the project navigator, select the MerchantApp project.
3. In the project editor, select the MerchantApp target, and click General.
4. In the Bundle Identifier field in the Identity section, enter your MerchantApp App ID.



Add Your Merchant Identifier to the MerchantApp Project

To add your Apple Pay merchant identifier to the MerchantApp project:

1. Obtain a Stripe Apple Pay merchant certificate, following the instructions in Chapter 1 of the book.
2. Open the MerchantApp project in `<red_project_dir>/client_app/MerchantApp.xcodeproj`.
3. Turn on the Apple Pay capability in the project, as described in Chapter 1.

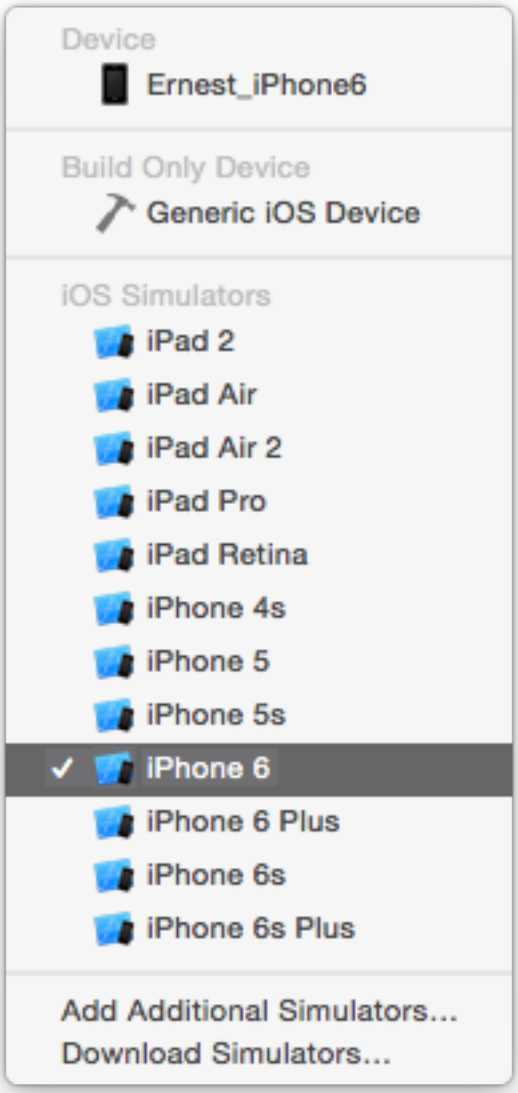
Add Your Payment Gateway's Keys to the MerchantApp Project

The `private_keys.h` file in the MerchantApp project contains the publishable keys your payment gateway provides. Add appropriate constants (and their values) to this file, to keep them in one place. In this case, set the `StripePublishableKey` constant to the test key provided by Stripe. Also, set constants and values for the Apple Pay merchant identifiers used in your app.

Run MerchantApp in an iOS Simulator or Device

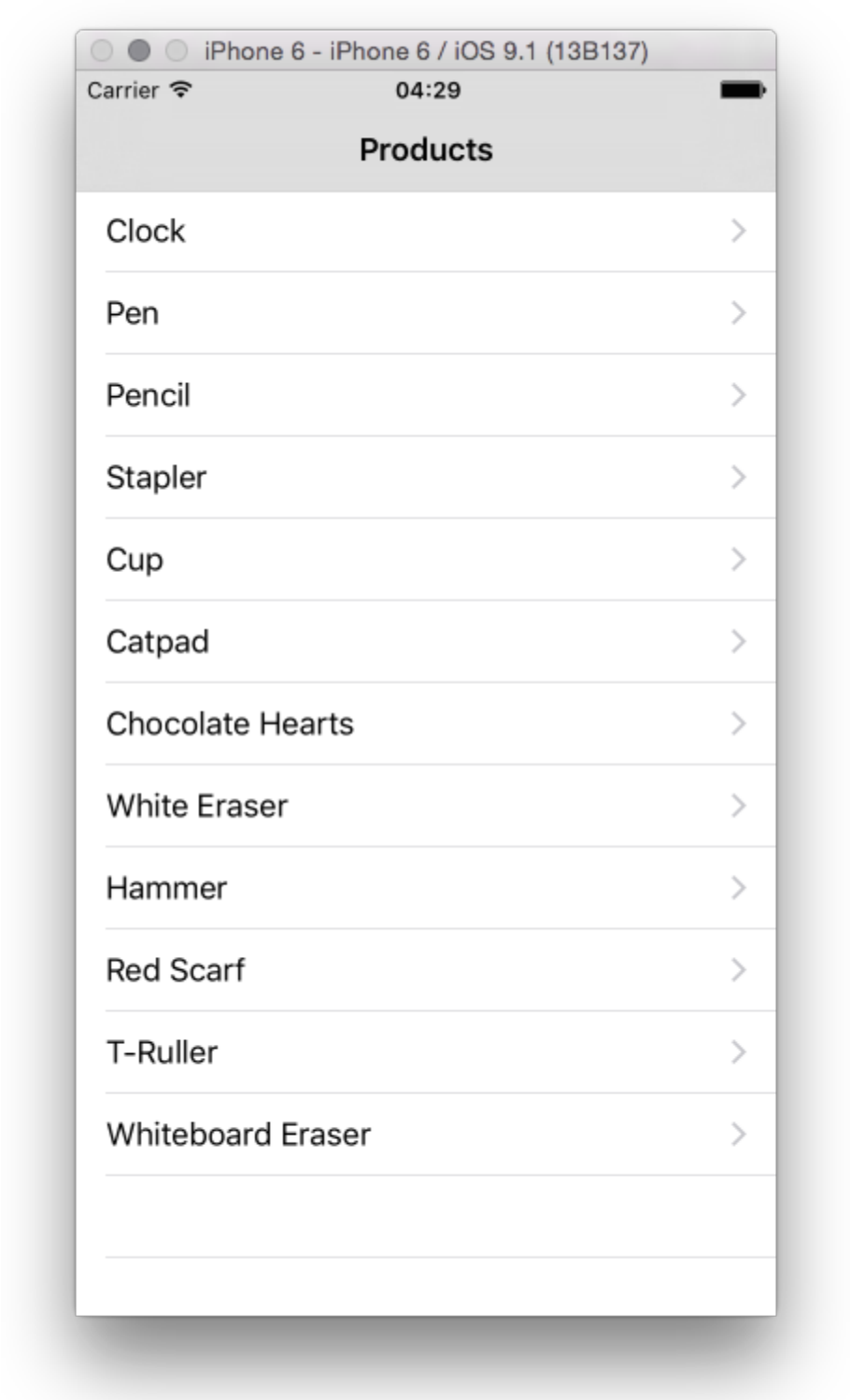
To run MerchantApp in an iOS simulator:

1. Open the MerchantApp project in Xcode.
2. Select the a simulator from the scheme toolbar menu.



3. Choose Product > Run.

The MerchantApp should be running in an iOS simulator, as shown here:



To run MerchantApp on your development iOS device:

1. Connect your iOS device to your development Mac using a USB cable.
2. Select your device from the Devices group in the scheme toolbar menu.
3. Choose Product > Run.

Revision History

Version	Description
1 (2016-02-21)	First version of this readme file.