Apple Pay Essentials

Apple Pay, one of the most talked about offerings of the latest iOS 9 release, is a digital wallet and electronic payment system developed by Apple Inc. It allows payment to merchants, using Near Field Communication (NFC), and within iOS apps. Implementing Apple Pay within apps for payment is a bit tricky, but our book solves this problem for you.

Whether you are a brand new iOS app developer or a seasoned expert, this book arms you with the necessary skills to successfully implement Apple Pay. We start off by teaching you how to obtain the certificates necessary to encrypt customers' payment information. You will then learn how to determine whether the customer can use Apple Pay, and how to create payment requests. You will come to grips with designing a payment-processor program to interact with the payment gateway. By the end of this book, you will be able to build a fully functional Apple Pay-integrated iOS app.

Who this book is written for

This book is for anyone who wants to integrate Apple Pay in their applications. Basic familiarity with programming and the Xcode developer tools is expected.

What you will learn from this book

- Design a product card that includes the Apple Pay button
- Implement the Apple Pay workflow in an efficient way
- Use NSDecimalNumber objects to perform financial calculations accurately
- Manage custom order information in the Apple Pay workflow and your custom order management system
- Extract payment information from a payment token
- Implement a secure, server-side payment-processor program
- Find out everything you wanted to know about Apple Pay in iOS 9

Harness the power of Apple Pay in your iOS apps and integrate it with global payment gateways

Ernest Bruce
In this package, you will find:

- The author biography
- A preview chapter from the book, Chapter 1 'Getting Started with Apple Pay'
- A synopsis of the book’s content
- More information on Apple Pay Essentials
Ernest Bruce is an accomplished technical writer and software engineer. He has worked for 13 years at Apple, Inc., where he held the position of Senior Technical Writer writing developer documentation. At Apple, Ernest specialized in writing documentation for the Xcode toolset, focusing on the Xcode user guide, and Xcode overview, as well as unit testing documentation and sample code. He also worked extensively on API documentation for the OS X and iOS platforms. Ernest helped design and develop the Xcode help articles that aid developers in getting around the user interface of the Xcode app. Before his years at Apple, Ernest worked as a programmer for Ping, Inc., where he helped manage the manufacturing processes using APL (A Programming Language). Ernest also has extensive experience in customer service, which has been instrumental to him developing content that readers find clear and easy to read, and that makes complex concepts more approachable.

Ernest is the head of Nerd Brawn, LLC, a software development company that focuses on developing platforms that help people learn about their environment and each other in innovative ways. The company is also working on new techniques to present content on desktop computers, tablets, and mobile phones.
Whether you are relatively new to iOS app development or a seasoned expert, *Apple Pay Essentials* provides the skills that you need to easily incorporate Apple Pay into the payment workflows of your apps. This book shows you how to obtain the certificates that ensure payment information is securely transmitted between the user's iOS device, your payment gateway, and the banks involved in an Apple Pay transaction. The book teaches you how to provide a simple and consistent user experience, which expedites the time between desire and acquire. You learn how to respond to changes that the user makes to your preconfigured payment sheet, which is where the user confirms or modifies order and payment details. This book guides you through the interactions that your app makes with your payment gateway and your order-processing system. Finally, this book shows you how to design a simple order-processing web app that processes orders and payments submitted by client apps.

**What this book covers**

*Chapter 1, Getting Started with Apple Pay,* describes how online payments work in general, and it introduces Apple Pay—a simpler and more secure online payment model. This chapter also shows you how to obtain the Apple Pay merchant certificates that ensure that only the appropriate entities have access to sensitive payment information.

*Chapter 2, Payment Request Workflow,* describes how to create a payment request, which is an object that stores information that is critical to payment processing (such as currency and payment network requirements) and details about the customer's order. It also shows you how to manage the main elements of the Apple Pay user experience: the *Apple Pay* button, and the payment sheet. This chapter also explains how to get inventory details from an order-processing web app, and how to present product information to the user.
Chapter 3, Payment Authorization Workflow, shows you how your app should respond to payment sheet events, such as shipping address change, and user authorization of the payment.

Chapter 4, Payment Processing Workflow, describes the actors and operations involved in processing a payment, including your order processing web app.

Chapter 5, Designing an Order Management Server, describes the main components of an order management server, including its data structure, and client API.

Chapter 6, Apple Pay API Summary, summarizes the API that is used for Apple Pay transactions.
Getting Started with Apple Pay

Apple Pay is a mobile payment system that lets iPhone users pay for goods and services using Touch ID. Instead of entering or confirming payment card information (credit or debit card) every time they make a purchase, users can authorize payment for items securely by touching the Home button. It is important to note that during an Apple Pay transaction, payment card information never leaves the user's phone; this information is stored securely in the device. Instead, a payment token stores all the information you need to process the payment all the way from authorization to settlement (that is, when the user's funds are transferred to your merchant bank account).

Using Apple Pay, you do not have to store your customers' payment card information on your servers. This helps reduce your customers' misgivings about paying for goods within your app; they trust that their payment card information is secure in their devices. You benefit by not having to deal with payment card information at all, at least not for Apple Pay-based transactions. (When a user's device does not support Apple Pay, or the user has not yet added payment cards to the device, you may have to process payment using regular means, which may involve capturing and storing payment card information.)

Although you are freed from storing payment card details on your systems, you still have to deal with processing the payments, either directly or through a payment gateway. In either case, you need to get an Apple Pay merchant identifier and certificate to decrypt the payment token that Apple Pay creates with a transaction's payment information. To use Apple Pay in your app, you need to enable the Apple Pay capability in your project, which requires the Apple Pay merchant identifier.
This chapter describes how online payments work, online payments being a web-centric version of the traditional **Electronic Draft Capture (EDC)** system used to process credit card transactions. You will also learn the basics of the Apple Pay payment workflow, starting with displaying the Apple Pay button when Apple Pay is available on the user’s device, presenting the Apple Pay payment sheet, and processing the transaction on your servers.

This chapter will do the following:

- Provide an overview of the online payment process
- Introduce the Apple Pay payment workflow
- Show you how to create an Apple Pay merchant identifier and certificate
- Describe how to turn on the Apple Pay capability for an app in Xcode

### An overview of the online payment process

Customers usually carry payment cards (debit or credit cards) in purses or wallets, which they use to pay for goods and services. When a cardholder pays a merchant with a payment card, the merchant usually uses a payment gateway to process the payment. A payment gateway is an e-commerce service that authorizes payment card-based transactions. The payment gateway performs several tasks to process the transaction, but its main task is the encryption of payment card information before submitting the transaction for authorization to a payment processor. A payment processor interacts with the bank that issued the customer's card (known as the **issuing bank** or **issuer**) that ultimately authorizes or declines the transaction. The payment processor may be implemented by the payment gateway, a third party, or the merchant. A merchant would implement a custom payment processor to, for example, integrate with a custom inventory and ordering system.

Merchants that do not manage inventory may deal only with a payment gateway. Payment gateways provide libraries or frameworks that apps can link to. When processing a payment, the app hands off a payment token to the library, which processes the payment and returns the result (authorized or declined) to the app. The gateway performs all the tasks necessary to authorize the transaction and transfer the payment amount from the card issuer to the merchant's acquiring bank. The **acquiring bank** (also known as the **acquirer**) is the bank that receives the cardholder's payments and credits them to the merchant's bank account (which is a special type of account used to receive payment from payment cards, also known as a merchant account).
Merchants that need to integrate with custom ordering and inventory management systems need a more hands-on approach to payment processing. This is the scenario discussed in this book.

First, let's talk about how online payment systems work. The payment process takes place in two phases:

- Authorization
- Settlement

In a successful authorization, an *authorization hold* is placed on the customer's card, reserving the funds that finance the transaction. Later, the merchant consumes or settles the transaction to transfer the funds from the customer's card into the merchant's account.

The following steps describe the authorization process:

1. The customer presents a payment card to pay for a product or service.
2. The merchant encrypts the card's information and sends an authorization request to the payment gateway.
3. The payment gateway then forwards the authorization request to the payment processor.
4. The payment processor forwards the authorization request to the appropriate payment card association (Visa, MasterCard, American Express, Discover, and so on).
5. The card association forwards the authorization request to the issuing bank, which ultimately approves or declines the transaction. Some card associations, such as Discover and American Express, are also issuing banks.
6. The issuing bank receives the authorization request from the payment processor and sends its response (*authorized* or *declined*) to the payment processor. The issuing bank then holds a *transaction authorization* or authorization hold that links the merchant, payment card, and amount approved (the funds are reserved but not debited from the cardholder's account).
7. The payment processor forwards the issuing bank's response to the payment gateway.
8. The payment gateway, in turn, forwards the response to the merchant, who relays the information to the cardholder.
Either immediately, or at the end of the day, the merchant starts the settlement process to receive the funds. This process is similar to the procedure used to request the payment authorization; however, instead of authorizing the transaction, the issuing bank moves the authorization hold to a debit and prepares the transaction for settlement with the acquiring bank:

1. The merchant submits the approved authorization to its acquiring bank through the payment processor.
2. The acquiring bank makes a settlement request to the issuing bank.
3. The issuing bank makes a settlement payment to the acquiring bank.
4. The acquiring bank deposits the approved amount into the merchant's bank account.

The Apple Pay payment workflow

If you develop an app that is capable of interacting with a payment gateway to process payment cards, you or your company is a merchant, and the app is a merchant app.

This is an overview of the payment workflow:

1. **Present the Apple Pay button**: Present this button only if the user can make Apple Pay payments.
2. **Create the payment request**: This request contains essential payment information and details about the order.
3. **Present the payment sheet**: This sheet presents order information that the user can modify, such as shipping information.
4. **Respond to changes by the user**: As the user makes changes, update items such as shipping costs and discounts.
5. **Submit payment information to payment gateway**: When the user authorizes the payment request, submit the payment and order information to the appropriate systems.
Presenting the Apple Pay button

When a user reaches a screen in your app that lets the user purchase something, the app should present the Apple Pay button (if the user can use Apple Pay on the device) so that the user can tap the button, verify the purchase details, and authorize the app through Touch ID to complete the order and charge the order amount to the appropriate payment card. Deciding whether the user can use Apple Pay involves two steps:

- Determining whether the device supports Apple Pay
- Determining whether the user has added payment cards that you support to the device

Your app must make both checks before displaying the Apple Pay button. If either check fails, the app must not present the Apple Pay button. Instead, it should offer a traditional payment method (such as obtaining a credit card number and a shipping address) through a Buy button.

Creating the payment request

If the user can use Apple Pay, your app prepares a payment request. A payment request is an object that describes the items to charge for, the card associations that you support, and billing and shipping information.

The main components of a payment request are payment summary items, which describe the payment request to the user. A payment summary item represents a component of the transaction, such as the subtotal, a discount, shipping cost, tax, and the grand total. Each item has a label that describes what each amount means. The last item is the most important because it identifies the payee and the debit amount that the user will see in the next payment card statement. Therefore, this item should have your company’s name as its label.

In addition to the payment summary items, your app sets properties of the payment request that describe which card associations and online payment protocols you support. Your app must support at least the 3D Secure protocol. The EMV (Europay, MasterCard, and Visa) protocol is optional.
The payment request also lets you indicate that you want the user to specify particular order details, such as shipping or billing information. For example, you may require an e-mail or postal address.

If your ordering system requires additional information, such as the order number, you can include this information in the payment request as custom application data. Apple Pay includes a hash of this information in the payment token you receive when the user authorizes the payment. If your ordering system requires this information later, your app must be able to provide it separately.

**Presenting the payment sheet**

Once your app creates the payment request and the user taps the Apple Pay button, the app presents a payment sheet to the user. The payment sheet (formally known as the payment authorization view controller) presents the payment summary items in the payment request to the user for review. The user can change aspects of the order before authorizing payment. The user may also decide not to purchase the goods and cancel the transaction.

**Responding to order changes and payment authorization**

Your app implements a delegate of the payment sheet to respond to the user's actions by, for example, updating the order shipping cost and grand total when the user chooses a different shipping method.

When the user authorizes the payment request with Touch ID, Apple Pay interacts with the device's secure element (the chip that securely stores payment card details on the device, details that not even Apple has access to) and Apple's servers to generate a one-time-use payment token. The payment information describes the payment transaction and contains all the information needed to charge the payment amount to the user's payment card (but this does not contain card numbers). Apple encrypts the information in the token on its servers using your merchant certificate.
Submitting the payment information to the payment gateway

When the payment sheet tells its delegate that the user has authorized the payment request and sends the user the payment information, the delegate calls a synchronous method that forwards the payment information to your payment gateway. When the method returns, it provides the delegate with the result of the payment request. If the payment request is approved, the payment sheet displays a confirmation to the user that the transaction is approved and informs its delegate. The delegate then dismisses the payment sheet and displays a custom confirmation screen; such a screen may display the order number and a thank you message. If the payment request is not approved, the delegate must display an appropriate screen and ask the user for another form of payment.

Enabling Apple Pay in your app

For your app to be able to use Apple Pay, you must have an Apple merchant identifier and merchant certificate. Apple uses the certificate to encrypt payment information in the payment token. Your payment gateway (Stripe, Worldpay, and so on) uses the certificate to decrypt information in the payment token.

Creating your app's Apple Pay merchant identifier

You must have access to your team's Member Center portal and your payment gateway's certificate management facilities.

Create your merchant identifier in your team's Member Center page through the following steps:

1. In Member Center, click on Certificates, Identifiers & Profiles.
2. Under iOS Apps, click on Identifiers.
3. Under Identifiers, click on Merchant IDs.
4. Click on Continue (if this is your first merchant identifier) or on the plus sign (+) button in the upper-right corner of the page.
5. Enter a description for the merchant identifier in the **Description** field, for example `MerchantApp merchant identifier`.

6. Enter the identifier string in the **ID** field, for example `merchant.com.company.merchantapp`.

7. Click on **Continue** and then click on **Register**.

8. Click on **Done**.

**Request an Apple Pay certificate from your payment gateway by performing the following:**

1. In your payment gateway’s certificate management page, create an Apple Pay certificate.

2. Download the **Certificate Signing Request (CSR)** file to your Mac.

**Now, follow these steps to create your app merchant certificate in Member Center:**

1. In the **Certificates, Identifiers & Profiles** page, under **iOS Apps**, under **Certificates**, click on **All**.

2. Then, select **Apple Pay Certificate** and click on **Continue**.

3. Under **Which Merchant ID would you like to use?**, select the appropriate merchant identifier and click on **Continue**.

4. Under **Generate your certificate**, click on **Choose File**.

5. Choose the CSR file you obtained from your payment gateway.

6. Next, click on **Generate** and then click on **Download** to download your app merchant certificate to your Mac.

**Upload your app merchant certificate to your payment gateway via the following steps:**

1. In your payment gateway's certificate management page, upload the merchant certificate you downloaded from the **Member Center** portal.

2. Confirm that your merchant certificate is listed in your payment gateway account.

**Installing your app's Apple Pay merchant certificate on your Mac**

Double-click on the merchant certificate you downloaded earlier from **Member Center**. **Keychain Access** will then open and install the certificate along with your other certificates.
Enabling Apple Pay in your app's Xcode project

To provide your app with access to Apple Pay, you need to turn on the Apple Pay capability in the Xcode project. Perform the following:

1. First, open your project in Xcode.
2. Select the target that builds the app to open the target editor.
3. Then, click on Capabilities.
4. Find the Apple Pay capability and toggle the corresponding switch to its on position.
5. In the dialog that appears, select the appropriate development team, and click on Choose.

|xcode-turn_on_apple_pay-done.png|
Summary
In this chapter, you learned about the online payment process followed by merchants to obtain card-based payments. The chapter introduced general online payment concepts to describe how an app uses Apple Pay to perform a similar function but more securely. Finally, you learned how to create the Apple Pay merchant identifier and merchant certificate to enable Apple Pay payment in your apps.

The next chapter focuses on the payment request workflow, where you present the Apple Pay button when Apple Pay is available on the device, create the payment request, and present the payment sheet based on that request.
Where to buy this book
You can buy Apple Pay Essentials from the Packt Publishing website.
Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.
Click here for ordering and shipping details.