**In-App Purchase Programming Guide**

**应用内购买编程指南**

<http://developer.apple.com/library/ios/#documentation/NetworkingInternet/Conceptual/StoreKitGuide/Introduction/Introduction.html>

# Introduction

# 简介

In-App Purchase allows you to embed a store directly within your application. You implement In-App Purchase in your application using the Store Kit framework. Store Kit connects to the App Store on your application's behalf to securely process payments from the user. Store Kit prompts the user to authorize the payment, then notifies your application so that it can provide items the user purchased. You can use this in-application payment functionality to collect payment for enhanced functionality or additional content usable by your application.

应用内购买直接在您的应用里面嵌入了一个商店。您可以使用Store Kit框架在您的应用内实现IAP。Store Kit代替您的应用与App Store连接，安全地处理用户的支付。Store Kit会先提示用户授权支付，然后通知您的引用可以提供用户所购买的项目。您可以使用这种应用内的支付功能通过您的应用确认付款以增强功能或者增加可用内容。

For example, you could use In-App Purchase to implement any of the following scenarios:

* A basic version of your application with additional premium features.
* A book reader application that allows the user to purchase and download new books.
* A game that offers new environments (levels) to explore.
* An online game that allows the player to purchase virtual property.

例如，您可以可能在以下情景中用到IAP：

* 带有附加高级功能的基本版应用。
* 允许用户购买并下载新书的书籍阅读应用。
* 提供新的探险环境或关卡的游戏。
* 允许用户购买虚拟属性的在线游戏。

**Important** In-App Purchase only collects payment. You must provide any additional functionality, including unlocking built-in features or downloading content from your own servers. This documentation details the technical requirements of adding a store to your application. For more information on the business requirements of using In-App Purchase, see the [App Store Resource Center](http://developer.apple.com/appstore/). You must also read your licensing agreement for the definitive treatment of what you may sell and how you are required to provide those products in your application.

**重要**的是IAP仅仅用于确认支付。您必须自己提供所有附加功能，包括解锁内置功能或者从您的服务器下载内容。本文档详述了在您的应用中添加商店的技术要求。至于使用IAP的业务要求等更多信息，请看[App Store Resource Center](http://developer.apple.com/appstore/" \t "_top)。您也必须阅读关于您要卖什么和您需要如何在您的应用中提供这些产品的许可协议。

## Who Should Read This Document

## 谁应该阅读本文档

You should read this if you are interested in offering additional paid functionality to users from within your application.

如果您对在您的应用中向用户提供附加的支付功能感兴趣，您应该阅读本文档。

## Organization of This Document

## 本文档的结构

This document contains the following chapters:

* [“Overview of In-App Purchase”](http://developer.apple.com/library/ios/documentation/NetworkingInternet/Conceptual/StoreKitGuide/APIOverview/OverviewoftheStoreKitAPI.html#//apple_ref/doc/uid/TP40008267-CH100-SW1) introduces the functionality offered by In-App Purchase.
* [“Retrieving Product Information”](http://developer.apple.com/library/ios/documentation/NetworkingInternet/Conceptual/StoreKitGuide/RetrievingStoreInformation/RetrievingStoreInformation.html#//apple_ref/doc/uid/TP40008267-CH2-SW1) describes how your application retrieves information from the App Store about products it offers.
* [“Making a Purchase”](http://developer.apple.com/library/ios/documentation/NetworkingInternet/Conceptual/StoreKitGuide/MakingaPurchase/MakingaPurchase.html#//apple_ref/doc/uid/TP40008267-CH3-SW1) explains how your application requests payment from the App Store.
* [“Adding a Store to Your Application”](http://developer.apple.com/library/ios/documentation/NetworkingInternet/Conceptual/StoreKitGuide/AddingaStoretoYourApplication/AddingaStoretoYourApplication.html#//apple_ref/doc/uid/TP40008267-CH101-SW1) is a walkthrough that describes how to add a store to your application.
* [“Verifying Store Receipts”](http://developer.apple.com/library/ios/documentation/NetworkingInternet/Conceptual/StoreKitGuide/VerifyingStoreReceipts/VerifyingStoreReceipts.html#//apple_ref/doc/uid/TP40008267-CH104-SW1) describes how your server can verify that a receipt came from the App Store.
* [“Testing a Store”](http://developer.apple.com/library/ios/documentation/NetworkingInternet/Conceptual/StoreKitGuide/DevelopingwithStoreKit/DevelopingwithStoreKit.html#//apple_ref/doc/uid/TP40008267-CH103-SW1) discusses how to use the sandbox environment to test your application.
* [“Auto-Renewable Subscriptions”](http://developer.apple.com/library/ios/documentation/NetworkingInternet/Conceptual/StoreKitGuide/RenewableSubscriptions/RenewableSubscriptions.html#//apple_ref/doc/uid/TP40008267-CH4-SW2) describes how your application can implement subscriptions using the Apple Store to manage the renewal process for you.

本文档那个包括以下章节：

* [《IAP概述》](http://developer.apple.com/library/ios/documentation/NetworkingInternet/Conceptual/StoreKitGuide/APIOverview/OverviewoftheStoreKitAPI.html#//apple_ref/doc/uid/TP40008267-CH100-SW1)介绍了IAP所提供的功能。
* [《检索产品信息》](http://developer.apple.com/library/ios/documentation/NetworkingInternet/Conceptual/StoreKitGuide/RetrievingStoreInformation/RetrievingStoreInformation.html#//apple_ref/doc/uid/TP40008267-CH2-SW1) 介绍了您的应用如何从App Store检索它提供的产品信息。
* [《作出购买》](http://developer.apple.com/library/ios/documentation/NetworkingInternet/Conceptual/StoreKitGuide/MakingaPurchase/MakingaPurchase.html#//apple_ref/doc/uid/TP40008267-CH3-SW1) 介绍了您的应用如何从App Store请求支付。
* [《在您的应用中添加商店](http://developer.apple.com/library/ios/documentation/NetworkingInternet/Conceptual/StoreKitGuide/AddingaStoretoYourApplication/AddingaStoretoYourApplication.html#//apple_ref/doc/uid/TP40008267-CH101-SW1)》一个介绍如何在您的应用中添加商店的演练（例子）。
* [《验证商店收据](http://developer.apple.com/library/ios/documentation/NetworkingInternet/Conceptual/StoreKitGuide/VerifyingStoreReceipts/VerifyingStoreReceipts.html#//apple_ref/doc/uid/TP40008267-CH104-SW1)》 介绍了您的服务器如何验证来自App Store的收据。
* [《测试商店》](http://developer.apple.com/library/ios/documentation/NetworkingInternet/Conceptual/StoreKitGuide/DevelopingwithStoreKit/DevelopingwithStoreKit.html#//apple_ref/doc/uid/TP40008267-CH103-SW1)探讨了如何使用沙盒环境来测试您的应用。
* [《自动更新订阅》](http://developer.apple.com/library/ios/documentation/NetworkingInternet/Conceptual/StoreKitGuide/RenewableSubscriptions/RenewableSubscriptions.html#//apple_ref/doc/uid/TP40008267-CH4-SW2) 介绍了您的应用程序如何使用App Store为您管理更新过程以实现订阅。

## See Also

## 参见

The [App Store Resource Center](http://developer.apple.com/appstore/) describes the business side of using In-App Purchase, as well as the steps you need to take to sell a product within your application.

The [*iTunes Connect Developer Guide*](http://developer.apple.com/library/ios/documentation/LanguagesUtilities/Conceptual/iTunesConnect_Guide/1_Introduction/Introduction.html#//apple_ref/doc/uid/TP40011225) describes how to configure products and test accounts on the App Store.

The [*Store Kit Framework Reference*](http://developer.apple.com/library/ios/documentation/StoreKit/Reference/StoreKit_Collection/_index.html#//apple_ref/doc/uid/TP40008300) describes the API for interacting with the App Store.

[App Store Resource Center](http://developer.apple.com/appstore/)介绍了使用IAP的商业方面，以及在您的应用中卖出产品所需要采取的步骤。

[*iTunes Connect Developer Guide*](http://developer.apple.com/library/ios/documentation/LanguagesUtilities/Conceptual/iTunesConnect_Guide/1_Introduction/Introduction.html#//apple_ref/doc/uid/TP40011225) 介绍了如何在App Store注册产品和测试账户。

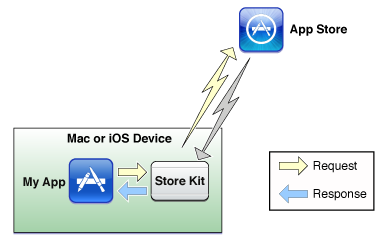
[*Store Kit Framework Reference*](http://developer.apple.com/library/ios/documentation/StoreKit/Reference/StoreKit_Collection/_index.html#//apple_ref/doc/uid/TP40008300) 介绍了与App Store的交互API。

# Overview of In-App Purchase

# IAP概述

Store Kit communicates with the App Store on behalf of your application. Your application uses Store Kit to receive localized information from the App Store about products you want to offer in your application. Your application displays this information to users and allows them to purchase items. When a user wants to purchase an item, your app calls Store Kit to collect payment from the user. Figure 1-1 shows the basic store model.

Store Kit代替您的应用与App Store交流。您的应用使用Store Kit接收来自App Store的关于您想要在您的应用中提供的产品的本地化信息。您的应用向用户显示这些信息，并允许他们购买某些项目。当用户想要购买某项目时，您的应用调用Store Kit确认来自用户的支付。图1-1展示了基本的商店模型。

**Figure 1-1** In-App Store model

The Store Kit API is only a small part of the process of adding a store to your application. You need to decide how to track the products you plan to deliver, how your application presents a store front to the user, and how your application delivers the products users purchase from your store. The rest of this chapter explains the process of creating products and adding a store to your application.

Store Kit API仅仅是在您的应用中添加商店的一个小步骤。您需要决定如何跟踪您计划提供的产品，您的应用如何将商店展示在用户面前，以及您的应用如何将用户从商店购买的产品提供给他们。本章的其余部分将介绍创建产品和在应用中添加商店的步骤。

## Products

## 产品

A **product** is any feature that you want to sell in your application’s store. Products are associated with the App Store through iTunes Connect in the same way that you create new applications. There are four supported kinds of products that you may sell using In-App Purchase:

产品可以是您想在您的应用商店中销售的任何功能。产品通过iTunes Connect与App Store相关联，关联方式与创建新应用是一样的。IAP支持以下4种产品用于销售：

* Content includes digital books, magazines, photos, artwork, game levels, game characters, and other digital content that can be delivered within your application.
* Functionality products unlock or expand features you’ve already delivered in your application. For example, you could ship a game with multiple smaller games that could be purchased by the user.
* Services allow your application to charge users for one-time services, such as voice transcription. Each time the service is used is a separate purchase.
* Subscriptions provide access to content or services on an extended basis. For example, your application might offer monthly access to financial information or to an online game portal.
* 内容 包括电子书，杂志，照片，艺术品，游戏关卡，游戏人物，以及其他可以在您的应用中提供的电子内容。
* 功能 产品解锁或者扩大您已经在应用中提供的功能。例如，您可以玩一个由多个小游戏组成的游戏，而这些小游戏则是由用户购买的。
* 服务 允许您的应用每次提供服务时向用户收费，例如语言转录。每次使用服务都是一次独立的购买。
* 订阅 在一个扩展的基础（功能）上提供对内容或服务的访问。例如，您的应用可能按月提供财务信息或在线游戏门户。

In-App Purchase provides a general mechanism for creating products, leaving the specifics of how your products are implemented up to you. However, there are few important guidelines to keep in mind as you design your application:

IAP提供了创建产品的通用机制，而将产品如何实现的细节留给您决定。但是，在您设计应用时，仍有少数重要的指导方针需要您牢记的：

* You must deliver a digital good or service within your application. Do not use In-App Purchase to sell real-world goods and services.
* You may not offer items that represent intermediary currency because it is important that users know the specific good or service they are buying.
* Items you offer for purchase may not contain, or relate to, pornography, hate speech, defamation, or gambling (simulated gambling is acceptable).
* 您必须在您的应用中提供数字商品或服务。切勿使用IAP销售真实的商品或服务。
* 您不能提供中介货币购买项，因为用户必须知道他们所购买的商品或服务的细节，这是很重要的。
* 您提供的购买项目不能包含或者涉及色情、偏见、诽谤或赌博（模拟赌博是可以接受的）。

For detailed information about what can be offered using In-App Purchase, consult your licensing agreement.

关于使用IAP能够提供什么的详细信息，请参看您的许可协议。

## Registering Products with the App Store

## 在App Store注册产品

Every product you wish to offer in your store must first be registered with the App Store through iTunes Connect. When you register a product, you provide a name, description, and pricing for your product, as well as other metadata used by the App Store and your application.

您希望在您的商店中提供的每一件产品都必须先通过iTunes Connect在App Store注册。注册产品时，您需要提供您的产品的名称、描述、产品价格以及App Store和您的应用要用到的其他数据信息。

You identify a particular product using a unique string called a **product identifier**. When your application uses Store Kit to communicate with the App Store, it uses product identifiers to retrieve the configuration data you provided for the product. Later, when a customer wants to purchase a product, your application identifies the product to be purchased using its product identifier.

使用一个独特的字符串标识一件特别的产品，称之为**产品标识**。当您的应用使用Store Kit与App Store通信时，会使用产品标识检索您为产品提供的配置数据。以后，有顾客想要购买产品时，您的应用也会使用其产品标识来标识将购买的产品。

The App Store supports many types of products:

App Store支持多种产品：

* **Consumable** products must be purchased each time the user needs that item. For example, one-time services are commonly implemented as consumable products.
* **消费型的产品在用户每次需要时都必须购买。例如，一次性服务通常作为消费型的产品。**
* **Non-consumable** products are purchased only once by a particular user. Once a non-consumable product is purchased, it is provided to all devices associated with that user’s iTunes account. Store Kit provides built-in support to restore non-consumable products on multiple devices.
* **非消费型的产品对于特定的用户来说仅购买一次。一旦购买了非消费型的产品，它将提供给所有与该用户iTunes帐号想关联的所有设备。**Store Kit提供了在多个设备上恢复非消费型产品的内置支持。
* **Auto-renewable subscriptions** are delivered to all of a user’s devices in the same way as non-consumable products. However, auto-renewable subscriptions differ in other ways. When you create an auto-renewable subscription in iTunes Connect, you choose the duration of the subscription. The App Store automatically renews the subscription each time its term expires. If the user chooses to not allow the subscription to be renewed, the user’s access to the subscription is revoked after the subscription expires. Your application is responsible for validating whether a subscription is currently active and can also receive an updated receipt for the most recent transaction.
* **自动更新订阅与非消费型产品一样也是提供给一个用户的所有设备。但是自动更新订阅在其他方面又有所不同。当您在iTunes Connect上创建一份自动更新订阅，您需要选择订阅的时长。每到这个时期一满，App Store就会自动的更新这个订阅。如果用户选择不允许更新订阅，用户对订阅的访问将会在订阅期满之后取消。您的应用程序负责验证订阅当前是否活动的，并且也可以收到最近的更新交易收据。**
* **Free subscriptions** are a way for you to put free subscription content in Newsstand. Once a user signs up for a free subscription, the content is available on all devices associated with the user’s Apple ID. Free subscriptions do not expire and can only be offered in Newsstand-enabled apps.
* **免费订阅是您在报摊提供免费订阅内容的途径。一旦用户注册了免费订阅，其内容在与该用户Apple ID关联的所有设备上都是可用的。免费订阅不会过期，但仅仅在报摊可用的应用中提供。**
* **Non-renewing subscriptions** are a mechanism for creating products with a limited duration. Non-renewing subscriptions differ from auto-renewable subscriptions in a few key ways:
  + The term of the subscription is not declared when you create the product in iTunes Connect; your application is responsible for providing this information to the user. In most cases, you would include the term of the subscription in the description of your product.
  + Non-renewing subscriptions may be purchased multiple times (like a consumable product) and are not automatically renewed by the App Store. You are responsible for implementing the renewal process inside your application. Specifically, your application must recognize when the subscription has expired and prompt the user to purchase the product again.
  + You are required to deliver non-renewing subscriptions to all devices owned by the user. Non-renewing subscriptions are not automatically synchronized to all devices by Store Kit; you must implement this infrastructure yourself. For example, most subscriptions are provided by an external server; your server would need to implement a mechanism to identify users and associate subscription purchases with the user who purchased them.
* **非更新的订阅** 是一种创建有限时长的产品的机制。非更新订阅在以下几个关键点上与自动更新订阅有所不同：
  + 在iTunes Connect创建产品时并没有声明订阅的时长；您的应用负责向用户提供这些信息。在多数情况下，您可以把将这个订阅期限包含在您的产品描述里。
  + 非更新的订阅可以被多次购买（就像消费型产品），并且不会被App Store自动更新。您负责在您的应用中实现更新步骤。特别地，您的应用必须确认订阅是否已过期并且提示用户再次购买该产品。
  + 您需要向用户拥有的所有设备提供非更新订阅。非更新的订阅不会被Store Kit自动同步到所有的设备；您必须自己实现这一基本功能。例如，大多说的订阅都是由外部服务器提供的；您的服务器需要实现一个机制，用于标识用户并将用户与其购买的应用关联起来。

Detailed information about registering products with the App Store can be found in [*iTunes Connect Developer Guide*](http://developer.apple.com/library/ios/documentation/LanguagesUtilities/Conceptual/iTunesConnect_Guide/1_Introduction/Introduction.html#//apple_ref/doc/uid/TP40011225).

在App Store中注册产品的详细信息可以在*[iTunes Connect Developer Guide](http://developer.apple.com/library/ios/documentation/LanguagesUtilities/Conceptual/iTunesConnect_Guide/1_Introduction/Introduction.html" \l "//apple_ref/doc/uid/TP40011225" \t "_self)*中找到。

## Feature Delivery

## 交付功能

The delivery mechanism your application uses to provide products to users has significant implications on its design and implementation. There are two basic models you should expect to use to deliver products to users: the built-in model and the server model. In both models, you track the list of products offered in the store and deliver products successfully purchased by users.

您的应用向用户提供产品的交付机制对它的设计和实现由重要的影响。您可能会使用以下两种基本的模型向用户提供产品：嵌入模型和服务器模型。在这两种模型中，您都需要追踪App Store提供的产品清单，并提供用户购买成功的产品。

### Built-in Product Model

### 嵌入模型

In the built-in product model, everything required to deliver products is built in to your application. This model is most often used to unlock functionality in your application. You could also use this model to deliver content provided in your application’s bundle. A key advantage of this model is that your application can promptly deliver products to the customer. Most built-in products should be non-consumable.

在嵌入产品模型中，您提供产品所需要的所有东西都嵌入在应用里。该模型大多数时候用于解锁应用中的功能。您也可以使用这种模型提供您的应用程序包中的内容。该模型的一个关键优势在于它可以及时的将产品提供给用户。大部分嵌入型产品都应该是非消费型产品。

**Important** In-App Purchase does not provide the capability for your application to be patched after a successful purchase. If your product requires changes to your application’s bundle, you must deliver an updated version of your application to the App Store.

重要：IAP不会在购买成功后向您的应用提供补丁功能。如果您的产品需要更改您的应用程序包，您必须向App Store提供应用程序的更新版本。

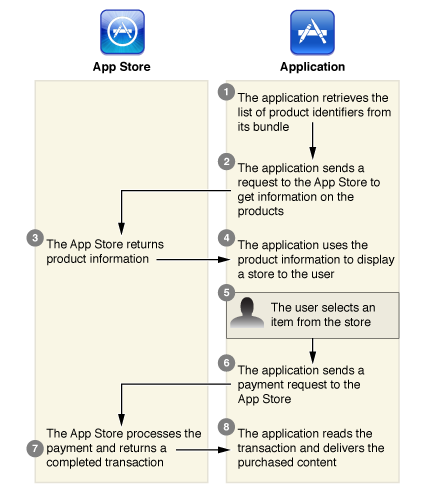
To identify products, your application stores the product identifiers in your application’s bundle. Apple recommends using a property list (plist) to track product identifiers for your built-in features. Content-driven applications can use this to add new content without modifying the source for your application.

为了识别产品，您的应用需要在应用程序包里存储产品标识。苹果建议使用一个属性列表为您的内置功能跟踪产品标识。内容导向的应用可以使用这种方式添加新的内容，而不用修改应用源。

After a product is successfully purchased, your application must unlock the feature and deliver it to the user. The simplest way to unlock features is by changing your application preferences. See [“Implementing Application Preferences”](http://developer.apple.com/library/ios/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/App-RelatedResources/App-RelatedResources.html#//apple_ref/doc/uid/TP40007072-CH6). Application preferences are backed up when users backs up their iOS-based devices. Your application may want to recommend to users that they back up their devices after making a purchase to ensure that purchases are not lost.

Figure 1-2 shows the series of actions your application takes to deliver a built-in product.

在产品被成功购买后，您的应用必须解锁功能并提供给用户。解锁功能最简单的方式是通过改变应用的偏好设置。请参考《[“实现应用偏好](http://developer.apple.com/library/ios/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/App-RelatedResources/App-RelatedResources.html" \l "//apple_ref/doc/uid/TP40007072-CH6" \t "_self)》。当用户备份其iOS设备时，应用偏好设置也会备份。您的应用应该推荐用户在购买后备份他们的设备，以确保购买不被丢失。图1-2展示了应用提供嵌入产品的一系列动作：

**Figure 1-2** Built-in product delivery

①应用程序从包中检索到产品标识列表。

②应用程序向App Store发送请求，获取产品相关信息。

③App Store返回产品信息。

④应用程序使用产品信息向用户展示一个商店。

⑤用户从商店选取一个购买项目。

⑥应用程序向App Store发送购买请求。

⑦App Store处理购买并返回完成的交易。

⑧应用程序阅读交易并交付购买的内容。

### Server Product Model

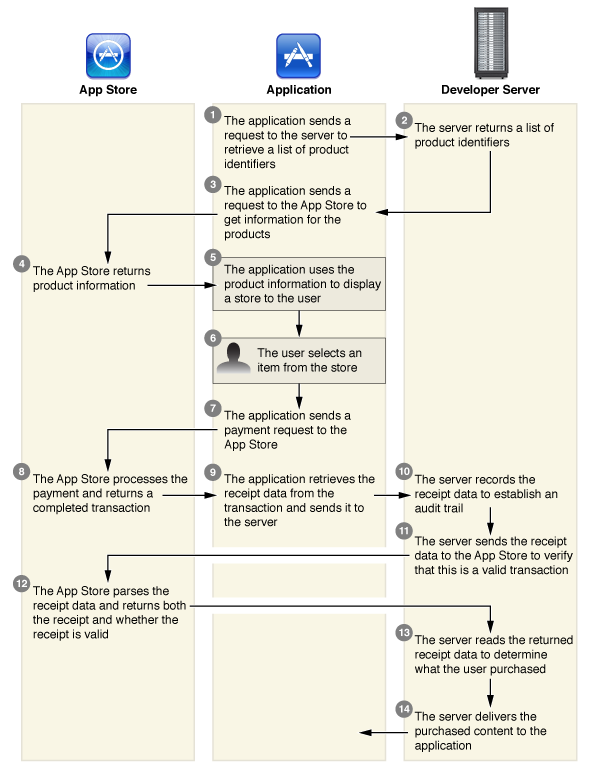
### 服务器产品模型

In the server product model, you provide a separate server that delivers products to your application. Server delivery is appropriate for subscriptions, services and content, because these products can be delivered as data without altering your application bundle. For example, a game might deliver new play environments (puzzles or levels) to the application. Store Kit does not define the design of your server or its interactions with your application. You are responsible for designing all interactions between your application and your server. Further, Store Kit does not provide a mechanism to identify a particular user. Your design may require you to provide a mechanism to identify a user. If your application requires these (for example, to track which subscriptions are associated with a particular user), you need to design and implement this yourself.

在服务器产品模型中，您提供一个分离的服务器，由服务器向您的应用交付产品。服务器交付适合于订阅、服务和内容（等类型的产品），因为这些产品可以以数据的形式交付，而不用改变您的应用程序包。例如，游戏类应用中可能交付新的环境（谜题或关卡）。Store Kit不会决定您应用程序的服务或交互的设计。您需要负责设计应用程序与服务器之间的所有交互。另外，Store Kit也不会提供对个别用用户的识别。您的设计需要自己提供一个机制识别用户。如果您的程序需要这些机制（例如，跟踪个别用户访问的订阅），您都需要自己来设计和实现。

Figure 1-3 expands the built-in model to show interactions with a server.

图1-3扩展了嵌入模型以展示与服务器的交互。

**Figure 1-3** Server product delivery

①应用向服务器发送检索产品标识列表的请求。

②服务器返回产品标识列表。

③应用程序向App Store发送请求，获取产品相关信息。

④App Store返回产品信息。

⑤应用程序使用产品信息向用户展示一个商店。

⑥用户从商店选取一个购买项目。

⑦应用程序向App Store发送购买请求。

⑧App Store处理购买并返回完成的交易。

⑨应用程序从交易中检索出收据数据并发给服务器。

服务器记录收据数据以建立一个审计跟踪。

服务器将收据数发送到App Store核实该交易是否有效。

App Store解析收据数据并返回收据和收据是否有效。

服务器阅读返回的收据数据以确定用户购买了什么。

服务器向应用程序交付购买的内容。

Apple recommends you retrieve product identifiers from your server, rather than including them in a property list. This gives you the flexibility to add new products without updating your application.

苹果推荐您从服务器检索产品标识，而不是将它们存在属性列表里。这样您可以灵活的添加新产品而不用更新应用程序。

In the server model, your application retrieves the signed receipt associated with a transaction and sends it to your server. Your server can then validate the receipt and decode it to determine which content to deliver to your application. This process is covered in detail in [“Verifying Store Receipts.”](http://developer.apple.com/library/ios/documentation/NetworkingInternet/Conceptual/StoreKitGuide/VerifyingStoreReceipts/VerifyingStoreReceipts.html#//apple_ref/doc/uid/TP40008267-CH104-SW1)

在服务器模型里，您的应用程序检索到交易相关的签名收据，并发送给您的服务器。您的服务器可以验证收据并对其解码以确定向您的应用支付何种内容。这个过程在[《验证收据》](http://developer.apple.com/library/ios/documentation/NetworkingInternet/Conceptual/StoreKitGuide/VerifyingStoreReceipts/VerifyingStoreReceipts.html" \l "//apple_ref/doc/uid/TP40008267-CH104-SW1)一文中做了详细的介绍。

The server model has additional security and reliability concerns. You should test the entire environment for security threats. [*Secure Coding Guide*](http://developer.apple.com/library/ios/documentation/Security/Conceptual/SecureCodingGuide/Introduction.html#//apple_ref/doc/uid/TP40002415) provides additional recommendations.

服务器模型具有额外的安全性和可靠性问题。你应该测试整个环境的安全威胁。*[《安全编码指南》](http://developer.apple.com/library/ios/documentation/Security/Conceptual/SecureCodingGuide/Introduction.html" \l "//apple_ref/doc/uid/TP40002415" \t "_self)* 提供了更多的建议。

Although non-consumable products may be recovered using the built-in capabilities of Store Kit, non-renewing subscriptions must be restored by your server. You are responsible for recording information about non-renewing subscriptions and restoring them to users. Optionally, consumable products could also be tracked by your server. For example, if your consumable product is a service provided by your server, you may want the user to retrieve the results of that request on multiple devices.

虽然非消费型产品可能使用Store Kit的嵌入功能恢复，但非更新的订阅必须通过服务器恢复。您需要负责记录非更新的订阅并且为您的用户恢复它们。可选地，您的服务器也可以跟踪消费型产品。例如，如果您的消费型产品是由您的服务器提供的一种服务，您可能希望得到用户在不同设备上请求的检索结果。

# Retrieving Product Information

# 检索产品信息

When your application is ready to display a store to the user, it must populate its user interface with information from the App Store. This chapter details how to request product details from the App Store.

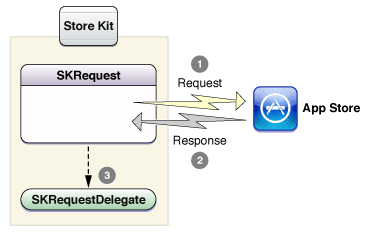
当您的应用准备向用户展示一个商店，该商店必须建立在用户界面之上，并且使用从App Store获取的信息。本章详细介绍了如何从App Store请求产品详情。

## Sending Requests to the App Store

## 向App Store发送请求

Store Kit provides a common mechanism to request information from the App Store. Your application creates and initializes a request object, attaches a delegate to it, and starts the request. Starting a request transmits it to the App Store, where it is processed. When the App Store processes the request, the request’s delegate is called asynchronously to deliver the results to your application. Figure 2-1 shows the request model.

Store Kit提供了从App Store请求信息的通用机制。您的应用程序创建并初始化一个请求对象，并给他附加一个委托，就可以开始这个请求。开始一个请求并发送到App Store进行处理。当App Store处理请求时，请求委托将被异步调用以向应用程序提供结果。图2-1展示了请求模型。

**Figure 2-1** Store Kit request model

If your application quits while a request is pending, your application needs to resend it.

如果应用程序在请求挂起时退出，应用程序则需要重新发送请求。

### SKRequest

[SKRequest](http://developer.apple.com/library/ios/documentation/StoreKit/Reference/SKRequest/Reference/Reference.html#//apple_ref/occ/cl/SKRequest) is an abstract base class for requests sent to the store.

[SKRequest](http://developer.apple.com/library/ios/documentation/StoreKit/Reference/SKRequest/Reference/Reference.html#//apple_ref/occ/cl/SKRequest)是一个发送请求到商店的抽象基类

### SKRequestDelegate

[SKRequestDelegate](http://developer.apple.com/library/ios/documentation/StoreKit/Reference/SKRequestDelegate/Reference/Reference.html#//apple_ref/occ/intf/SKRequestDelegate) is a protocol that your application implements to handle requests that completed successfully and requests that failed because of an error.

[SKRequestDelegate](http://developer.apple.com/library/ios/documentation/StoreKit/Reference/SKRequestDelegate/Reference/Reference.html#//apple_ref/occ/intf/SKRequestDelegate)是应用程序实现的协议，用于处理成功完成的请求和由于某些错误而失败的请求。

## Requesting Information About Products

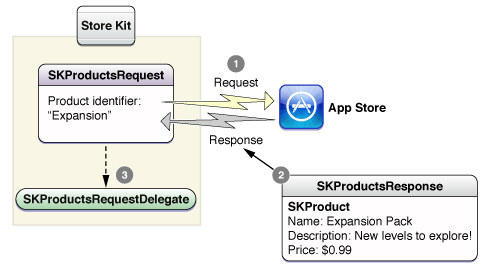
## 请求产品信息

Your application uses a **products request** to retrieve localized information about a product. Your application creates a request that contains a list of product identifier strings. As described earlier, your application might embed product identifiers inside your application or it might retrieve the identifiers from an external server.

应用程序使用产品请求检索产品的本地化信息。应用程序创建一个包含一系列产品标识字符串的请求。正如前面所述，应用程序可能将产品标识嵌入在程序里，或者从外部服务器检索标识。

When you start the products request, the product identifier strings are transmitted to the App Store. The App Store responds with the localized information you previously entered in iTunes Connect. You use these details to populate the user interface of your store. Figure 2-2 shows a products request.

当您开始产品请求，产品标识字符串将被发送到App Store。App Store会响应您预先在iTunes Connect中输入的本地化信息。您要使用这些信息来填充您的商店的用户界面。图2-2展示了一个产品请求。

**Figure 2-2** A request for localized product information

**Important** You must make a product request for a particular product identifier before allowing the user to purchase that product. Retrieving product information from the App Store ensures that you are using a valid product identifier for a product you have marked available for sale in iTunes Connect.

注意 您必须在用户购买产品之前为每一个产品标识发送一个产品请求。从App Store检索产品信息能够保证您正使用的产品标识，是在iTunes Connect中正在销售的可用产品标识。

### SKProductsRequest

An [SKProductsRequest](http://developer.apple.com/library/ios/documentation/StoreKit/Reference/SKProductsRequest/Reference/Reference.html#//apple_ref/occ/cl/SKProductsRequest) object is created with a list of product identifier strings for the products you want to display in your store.

[SKProductsRequest](http://developer.apple.com/library/ios/documentation/StoreKit/Reference/SKProductsRequest/Reference/Reference.html#//apple_ref/occ/cl/SKProductsRequest)对象通过一系列产品标识字符串创建，它们标识您想要在您的商店中显示的产品。

### SKProductsRequestDelegate

The [SKProductsRequestDelegate](http://developer.apple.com/library/ios/documentation/StoreKit/Reference/SKProductsRequestDelegate/Reference/Reference.html#//apple_ref/occ/intf/SKProductsRequestDelegate) protocol is implemented by an object in your application to receive the response from the store. It receives the response asynchronously when the request is successfully processed.

[SKProductsRequestDelegate](http://developer.apple.com/library/ios/documentation/StoreKit/Reference/SKProductsRequestDelegate/Reference/Reference.html#//apple_ref/occ/intf/SKProductsRequestDelegate)协议由应用程序里从商店接收相应的对象实现。当请求被成功处理时，它会异步接收响应。

### SKProductsResponse

An [SKProductsResponse](http://developer.apple.com/library/ios/documentation/StoreKit/Reference/SKProductsResponse/Reference/Reference.html#//apple_ref/occ/cl/SKProductsResponse) object contains a [SKProduct](http://developer.apple.com/library/ios/documentation/StoreKit/Reference/SKProduct_Reference/Reference/Reference.html#//apple_ref/occ/cl/SKProduct) object for each valid product identifier in the original request as well as a list of the product identifiers that were not recognized by the store. The store might not recognize the identifier for a number of reasons; it might be misspelled, marked unavailable for sale, or changes you have made in iTunes Connect have not propagated to all of the App Store servers.

[SKProductsResponse](http://developer.apple.com/library/ios/documentation/StoreKit/Reference/SKProductsResponse/Reference/Reference.html#//apple_ref/occ/cl/SKProductsResponse)对象为原始请求中每一个可用产品标识以及商店不可识别的一系列产品标识包含了一个[SKProduct](http://developer.apple.com/library/ios/documentation/StoreKit/Reference/SKProduct_Reference/Reference/Reference.html#//apple_ref/occ/cl/SKProduct)对象。商店可能由于诸多原因无法识别标识；可能是拼写错误，可能是被标记为不再出售，或者您在iTunes Connect中所做的修改还没有传播到所有的App Store服务器。

### SKProduct

An [SKProduct](http://developer.apple.com/library/ios/documentation/StoreKit/Reference/SKProduct_Reference/Reference/Reference.html#//apple_ref/occ/cl/SKProduct) object provides localized information about a product you’ve registered with the App Store.

[SKProduct](http://developer.apple.com/library/ios/documentation/StoreKit/Reference/SKProduct_Reference/Reference/Reference.html#//apple_ref/occ/cl/SKProduct)对象提供了您在App Store注册的产品的本地化信息。

# Making a Purchase

# 作出购买

When the user is ready to purchase an item, your application asks the App Store to collect payment. When your application asks for payment, the App Store creates a persistent transaction and continues to process the payment, even if the user quits and relaunches your application. The App Store synchronizes the list of pending transactions with your application and delivers updates to your application when the status of any of these transactions changes.

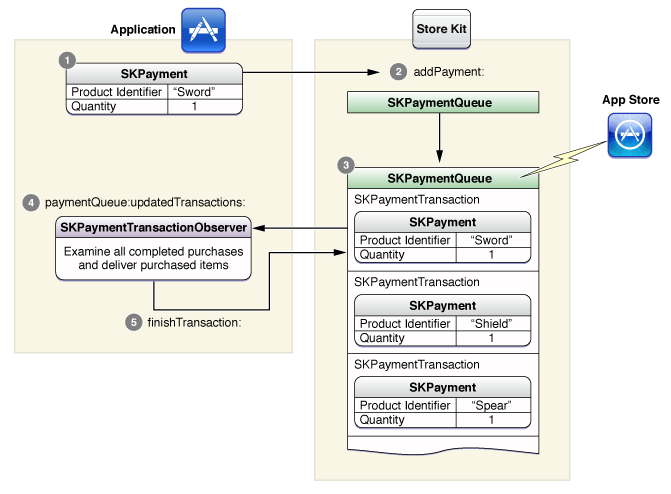
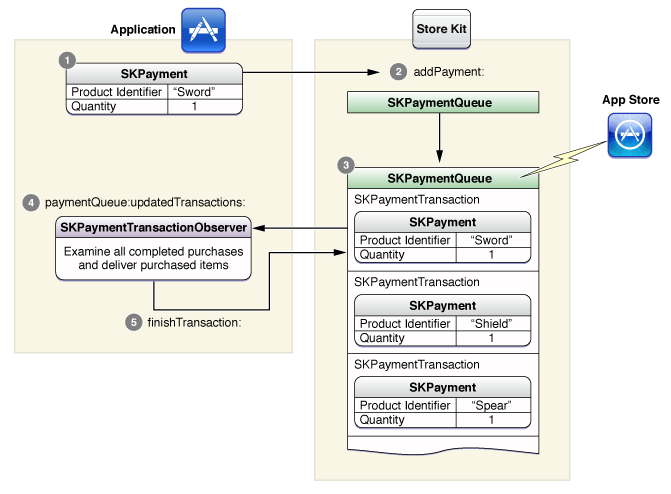
当用户准备购买物品，您的应用程序要求App Store收取款项。当应用程序要求付款，App Store创建一个持续存在的交易并继续处理付款，即使用户退出并重新启动你的应用。App Store同步您的应用程序的未完成交易列表，并且当这些交易的状态发生改变时向您的应用程序提供更新。

## Collecting Payments

## 收取款项

To collect payment, your application creates a payment object and queues it on the payment queue, as shown in Figure 3-1.

为了收取支付，应用程序会创建一个支付对象并将其排入支付队列，如图3-1所示。

**Figure 3-1** Adding a payment request to the queue

When the payment is added to the payment queue, a persistent transaction is created to hold it. After the payment is processed, the transaction is updated with information about the payment collection. Your application implements an observer that receives messages when transactions are updated. The observer should provide purchased items to the user and then remove the transaction from the payment queue.

当支付被添加到支付队列时，会创建一个持续存在的交易以控制它。在处理了这个支付之后，该交易对象会更新支付收取的相关信息。您的应用程序需要实现一个观察器，当交易更新时用于接收信息。观察器应该向用户提供购买物品，然后将交易从支付队列中移除。

### SKPayment

Collecting payment starts with a payment object. The payment object includes a product identifier and optionally includes the quantity of that product to be purchased. You can queue the same payment object more than once; each time a payment object is queued results in a separate request for payment.

收取款项有支付对象开始。支付对象必须包含产品标识，可选的包含被购买产品的数量。您可以多次为同一个支付对象排队；每次支付对象排队都会成为一个独立的支付请求。

Users can disable the ability to make purchases in the Settings application. Before attempting to queue a purchase, your application should first confirm that payment can be processed. You do this by calling the payment queue’s [canMakePayments](http://developer.apple.com/library/ios/documentation/StoreKit/Reference/SKPaymentQueue_Class/Reference/Reference.html#//apple_ref/occ/clm/SKPaymentQueue/canMakePayments) method.

用户可以在“设置”（指苹果的设置）中禁用购买功能。在试图排队购买之前，您的应用应该先确认支付能否被处理。您可以通过调用支付队列的[canMakePayments](http://developer.apple.com/library/ios/documentation/StoreKit/Reference/SKPaymentQueue_Class/Reference/Reference.html#//apple_ref/occ/clm/SKPaymentQueue/canMakePayments)方法完成这项工作。

### SKPaymentQueue

The payment queue is used to communicate with the App Store. When payments are added to the queue, Store Kit transmits the request to the App Store. Store Kit presents dialogs to ask the user to authorize payment. The completed transaction is returned to your application’s observer.

支付队列用于与App Store进行交流。当支付被添加到队列中，Store Kit将请求传送给App Store。Store Kit会展示出对话框让用户授权支付。已完成的交易会返回到您的应用的观察者（Observer）。

### SKPaymentTransaction

A transaction is created for every payment added to the queue. Each transaction has properties that allow your application to determine the status of the transaction. When payment is collected, the transaction includes additional details about the successful transaction.

每个支付都会创建交易，并加入到队列中。每个交易都具有允许您的应用确定交易状态的特性。当支付被收取，该交易就会包含成功交易的更多细节。

Although your application can ask the payment queue for a list of pending transactions, it is more common for an application to wait until the payment queue notifies the payment queue’s observer with a list of updated transactions.

尽管您的应用可以向交易队列请求一系列的待定交易，但更常见的情况是应用程序会等待，直到交易队列通知其观察者有已更新的交易。

### SKPaymentTransactionObserver

Your application implements the [SKPaymentTransactionObserver](http://developer.apple.com/library/ios/documentation/StoreKit/Reference/SKPaymentTransactionObserver_Protocol/Reference/Reference.html#//apple_ref/occ/intf/SKPaymentTransactionObserver) protocol on an object and adds it as an observer to the payment queue. The observer’s primary responsibility is to examine completed transactions, deliver items that were successfully purchased, and remove those transactions from the payment queue.

您的应用需要在某个对象中实现[SKPaymentTransactionObserver](http://developer.apple.com/library/ios/documentation/StoreKit/Reference/SKPaymentTransactionObserver_Protocol/Reference/Reference.html#//apple_ref/occ/intf/SKPaymentTransactionObserver)协议，并将其作为观察者加入到支付队列中去。观察者的主要职责是检查成功的交易，发布成功购买的项目，并将这些交易从支付队列中移除。

Your application should associate an observer with the payment queue when it launches, rather than wait until the user attempts to purchase an item. Transactions are not lost when an application terminates. The next time the application launches, Store Kit resumes processing transactions. Adding the observer during your application’s initialization ensures that all transactions are returned to your application.

您的应用程序应该在启动的时候就将观察者与支付队列关联起来，而不是等到用户试图购买时才去关联。即使应用程序终止交易也不会丢失。下次应用程序启动时，Store Kit会恢复对交易的处理。在您的应用程序初始化的时候就添加观察者能够确保所有的交易都返回到您的应用程序中。

## Restoring Transactions

## 恢复交易

Once a transaction has been processed and removed from the queue, your application normally never sees it again. However, if your application supports product types that must be restorable, you must include an interface that allows users to restore these purchases. This interface allows a user to add the product to other devices or, if the original device was wiped, to restore the transaction on the original device.

Store Kit provides built-in functionality to restore transactions for non-consumable products, auto-renewable subscriptions and free subscriptions. To restore transactions, your application calls the payment queue’s [restoreCompletedTransactions](http://developer.apple.com/library/ios/documentation/StoreKit/Reference/SKPaymentQueue_Class/Reference/Reference.html#//apple_ref/occ/instm/SKPaymentQueue/restoreCompletedTransactions) method. The payment queue sends a request to the App Store to restore the transactions. In return, the App Store generates a new restore transaction for each transaction that was previously completed. The restore transaction object’s [originalTransaction](http://developer.apple.com/library/ios/documentation/StoreKit/Reference/SKPaymentTransaction_Class/Reference/Reference.html#//apple_ref/occ/instp/SKPaymentTransaction/originalTransaction) property holds a copy of the original transaction. Your application processes a restore transaction by retrieving the original transaction and using it to unlock the purchased content. After Store Kit restores all the previous transactions, it notifies the payment queue observers by calling their [paymentQueueRestoreCompletedTransactionsFinished:](http://developer.apple.com/library/ios/documentation/StoreKit/Reference/SKPaymentTransactionObserver_Protocol/Reference/Reference.html#//apple_ref/occ/intfm/SKPaymentTransactionObserver/paymentQueueRestoreCompletedTransactionsFinished:) method.

If the user attempts to purchase a restorable product (instead of using the restore interface you implemented), the application receives a regular transaction for that item, not a restore transaction. However, the user is not charged again for that product. Your application should treat these transactions identically to those of the original transaction.

Non-renewing subscriptions and consumable products are not automatically restored by Store Kit. Non-renewing subscriptions must be restorable, however. To restore these products, you must record transactions on your own server when they are purchased and provide your own mechanism to restore those transactions to the user’s devices.