Sample Code

Implementing a store in your app using the StoreKit API

Offer In-App Purchases and manage entitlements using signed transactions and status information.

Download

iOS 26.0+ | iPadOS 26.0+ | Xcode 26.0+

Overview

The sample project demonstrates how to use StoreKit in a SwiftUI-based iOS app with a simulated app server, SKDemoServer. SKDemoServer is a Swift package that acts as the app's backend by vending product identifiers and persisting consumable purchase entitlements using SwiftData.

To test your implementation of In-App Purchases using StoreKit, you can use StoreKit Testing in Xcode and the sandbox environment. For more information, see Testing at all stages of development with Xcode and the sandbox, Testing In-App Purchases with sandbox, and Setting up StoreKit Testing in Xcode.

> **Note**
>
> This sample code project is associated with WWDC25 session 241: What's new in StoreKit and In-App Purchase.

Test the sample code project in Xcode

This sample code project implements StoreKit Testing in Xcode to let you test In-App Purchases without completing any product set up in App Store Connect. It defines In-App Purchases in a

`Products.storekit` file.

To test the sample in Xcode:

1. Select the sample target, then configure it to use your Developer team for signing. For more information, see Assign the project to a team in Preparing your app for distribution.
2. Edit the SKDemo "Run" scheme, and select `Products.storekit` for StoreKit configuration. For more information, see Enable StoreKit Testing in Xcode in Setting up StoreKit Testing in Xcode.
3. Build and run the sample app on a device or in the Simulator.
4. The sample app displays a list of products available for sale in `Products.storekit` upon launching. If the sample app fails to display an In-App Purchase, see TN3185: Troubleshooting In-App Purchases availability in Xcode.

Test the sample code project in the sandbox environment

To test your app in the sandbox environment, sign in to a Sandbox Apple Account. In the sandbox environment, you can test In-App Purchases using real product information from App Store Connect without incurring charges. This sample code project defines In-App Purchases in a `Products.plist` file. Before you can start testing the sample app in the sandbox environment, you need to complete configuration steps in App Store Connect and Xcode. For more information, see Prepare for sandbox testing in Testing In-App Purchases with sandbox.

To prepare this sample code project to run in the sandbox, perform the following steps in App Store Connect:

1. Find or create an app that supports In-App Purchases and you can use for testing purposes.
2. Select the app and make a note of its bundle ID.
3. Create an In-App Purchase that uses a product identifier from `Products.plist`. Use information from `Products.storekit` to set up the In-App Purchase. Repeat this step for all identifiers in `Products.plist`.
4. After you create the SKDemo+ subscription group, make note of its Group ID.
5. Create a Sandbox Apple Account.

To prepare this sample code project to use information from App Store Connect, perform the following steps in Xcode:

1. Select the sample target, then change the bundle ID to your testing app's bundle ID. For more information, see Set the bundle ID in Preparing your app for distribution.
2. Configure the target to use your Developer team for signing. For more information, see Assign the project to a team in Preparing your app for distribution.

3. Edit the SKDemo “Run” scheme, and remove `Products.storekit` from StoreKit configuration. For more information, see Disable StoreKit Testing in Xcode in Setting up StoreKit Testing in Xcode.

4. In the `Server.swift` file, replace 3F19ED53 with your new SKDemo+ subscription Group ID from App Store Connect.

You’re now ready to test this sample in the sandbox environment. Sign in to the device with a Sandbox Apple Account, then build and run this sample in Xcode. The sample app displays a list of products available for sale in the App Store upon launching. If the sample app fails to display an In-App Purchase, see TN3186: Troubleshooting In-App Purchases availability in the sandbox.

See Also

Product and subscription information

`struct Product`

Information about a product that you configure in App Store Connect.

`struct SubscriptionInfo`

Information about an auto-renewable subscription, such as its status, period, subscription group, and subscription offer details.

`typealias SubscriptionInfo`

Information about an auto-renewable subscription.

`typealias SubscriptionStatus`

Represents the renewal status information for an auto-renewable subscription.