API Collection

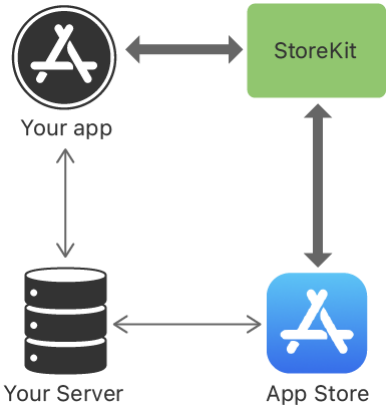
In-App Purchase

Offer content and services in your app across Apple platforms using a Swift-based interface.

Overview

With the In-App Purchase API, you can offer customers the opportunity to purchase digital content and services in your app. Customers can make the purchases within your app, and find your promoted products on the App Store.

The StoreKit framework connects to the App Store on your app's behalf to prompt for, and securely process, payments. The framework then notifies your app, making the transactions for In-App Purchases available to your app on all of the customer's devices. For each transaction that represents a current purchase, your app delivers the purchased products. To validate purchases, you can verify transactions on your server, or rely on StoreKit's verification.



The App Store can also communicate with your server. It notifies your server of transactions and auto-renewable subscription events through App Store Server Notifications, and provides the same transaction information, and more, through the App Store Server API.

To learn how adding In-App Purchases fits in an overall app development workflow for the App Store, see App Store Pathway. For an overview of In-App Purchases and its features, including its

configuration, testing capabilities, marketing for your products, and more, see Simple and safe In-App Purchases. For an overview on subscriptions, including creating subscription groups, Family Sharing, and more, see Auto-renewable subscriptions.

Configure In-App Purchases

To use the In-App Purchase API, you first need to configure the products that your app merchandises.

- In the early stages of development, you can configure the products in the StoreKit configuration file in Xcode, and test your code without any dependency on App Store Connect. For more information, see Setting up StoreKit Testing in Xcode.
- When you're ready for sandbox testing and production, configure the products in App Store Connect. You can add or remove products and refine or reconfigure existing products as you develop your app. For more information, see Configure In-App Purchase settings.

You can also offer apps and In-App Purchases that run on multiple platforms as a single purchase. For more information on universal purchase, see Add platforms.

Support a store in your app

The In-App Purchase API takes advantage of Swift features like concurrency to simplify your In-App Purchase workflows, and SwiftUI to build stores with StoreKit views. Use the API to manage access to content and subscriptions, receive App Store-signed transaction information, get the history of all In-App Purchase transactions, and more.

Related sessions from WWDC21

Session 10114: Meet StoreKit 2

The In-App Purchase API offers:

- Transaction information that's App Store-signed in JSON Web Signature (JWS) format.
- Transaction and subscription status information that's simple to parse in your app.
- An entitlements API, `currentEntitlements`, that simplifies determining entitlements to unlock content and services for your customers.

Related sessions from WWDC22

Session 110404: Implement proactive in-app purchase restore

To support a store in your app, implement the following functionality:

- Listen for transaction state changes using the transaction listener, `updates`, to provide up-to-date service and content while your app is running.
- Use StoreKit views to merchandise your products; or request products to display from the App Store with `products(for:)` and enable purchases using `purchase(options:)`. Unlock purchased content and services based on the purchase result, `Product.PurchaseResult`.
- Iterate through a customer's purchases anytime using the transaction sequence `all`, and unlock the purchased content and services.
- Optionally, validate the signed transactions and signed subscription status information that you receive from the API.

Topics

In-App Purchase merchandising

StoreKit views

Display a customizable In-App Purchase store using StoreKit views for SwiftUI.

Product and subscription information

{} Implementing a store in your app using the StoreKit API

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

Purchase requests and results

`struct PurchaseAction`

An action that starts an In-App Purchase.

`func purchase(options: Set<Product.PurchaseOption>) async throws -> Product.PurchaseResult`

Initiates a purchase for the product with the App Store and displays the confirmation sheet.

`enum PurchaseResult`

The result of a purchase.

Transaction history and entitlements

`struct Transaction`

Information that represents the customer's purchase of a product in your app.

`static var updates: Transaction.Transactions`

The asynchronous sequence that emits a transaction when the system creates or updates transactions that occur outside the app or on other devices.

`static var all: Transaction.Transactions`

A sequence that emits all the customer's transactions for your app.

`static var currentEntitlements: Transaction.Transactions`

A sequence of the latest transactions that entitle a customer to In-App Purchases and subscriptions.

JWS verification

`enum VerificationResult`

A type that describes the result of a StoreKit verification.

`enum VerificationError`

Error cases for StoreKit JWS verification.

Subscription status and renewal information

`struct Status`

The renewal status information for an auto-renewable subscription.

`struct RenewalInfo`

The renewal information for an auto-renewable subscription.

`typealias SubscriptionRenewalInfo`

Represents the renewal information for an auto-renewable subscription.

`struct RenewalState`

The renewal states of auto-renewable subscriptions.

`typealias SubscriptionRenewalState`

The renewal states of auto-renewable subscriptions.

`typealias SubscriptionPeriod`

Represents the duration of time between subscription renewals.

Subscription offers and offer codes

Supporting win-back offers in your app

Re-engage previous subscribers with a free or discounted offer for an auto-renewable subscription, for a specific duration.

Merchandising win-back offers in your app

Present win-back offers to eligible customers in your app with the win-back offer sheet or by implementing custom merchandising.

Supporting subscription offer codes in your app

Provide subscription service for customers who redeem offer codes through the App Store or within your app.

`struct SubscriptionOffer`

Information about a subscription offer that you configure in App Store Connect.

`struct OfferType`

The types of offers for auto-renewable subscriptions.

Promoted In-App Purchases

Supporting promoted In-App Purchases in your app

Display promoted In-App Purchases on your product page and handle purchases that users initiate on the App Store.

`struct PurchaseIntent`

An instance that emits purchase intents, which indicate that the customer initiated a purchase outside of your app, for your app to complete.

`struct PromotionInfo`

Information about a promoted In-App Purchase that customizes its order and visibility on the device.

Testing promoted In-App Purchases

Test your In-App Purchases before making your app available in the App Store.

App Store interactions

`enum AppStore`

Interactions with the App Store, such as managing subscriptions, verifying devices, authorizing payments, synchronizing transactions, getting the environment, and more.

`struct AppTransaction`

Information that represents the customer's purchase of the app, cryptographically signed by the App Store.

Storefront information

`struct Storefront`

The region and unique identifier of the App Store storefront for the device.

`static var current: Storefront?`

The current App Store storefront for product purchases.

`static var updates: Storefront.Storefronts`

The asynchronous sequence that emits storefront information when the system updates the storefront.

In-App Purchase Testing

Testing at all stages of development with Xcode and the sandbox

Verify your implementation of In-App Purchases by testing your code throughout its development.

- Testing In-App Purchases with sandbox

  Test your implementation of In-App Purchases using real product information and server-to-server transactions in the sandbox environment.

- Testing refund requests

  Test your app's implementation of refund requests, and your app's and server's handling of approved and declined refunds.

- Testing win-back offers in Xcode

  Validate your app's handling of win-back offers that you configure for the testing environment.

- Testing Ask to Buy in Xcode

  Validate your app's handling of Ask To Buy in the testing environment.

Advanced Commerce API interactions

`struct AdvancedCommerceProduct`

A product configured as a generic SKU in App Store Connect for use with the Advanced Commerce API.

- Sending Advanced Commerce API requests from your app

  Send Advanced Commerce API requests from your app that you authorize with a JSON Web Signature (JWS) you generate on your server.

- Generating JWS to sign App Store requests

  Create signed JSON Web Signature (JWS) strings on your server to authorize your API requests in your app.

Errors

`enum StoreKitError`

StoreKit In-App Purchase error codes.

Deprecated

- Choosing a StoreKit API for In-App Purchases

  Use the latest API to support In-App Purchases in new or existing apps, or the original API to support In-App Purchases in earlier operating systems.

Original API for In-App Purchase

Offer additional content and services in your app using the Original In-App Purchase API.

See Also

In-App Purchase

Understanding StoreKit workflows

Implement an in-app store with several product types, using StoreKit views.

Getting started with In-App Purchase using StoreKit views

Set up an in-app store using SwiftUI and StoreKit views.