

[StoreKit](#) / [In-App Purchase](#) / StoreKit views

API Collection

# StoreKit views

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

## Overview

The StoreKit views APIs provide UI to help you build a store for your In-App Purchases, and provide a way for customers to complete the purchase. The views support localization, so your customers see the product names, descriptions, and prices appropriate to their App Store storefront.

### Related session from WWDC23

Session 10013: [Meet StoreKit for SwiftUI](#)

StoreKit manages the layouts across all platforms, so the views look great on any device. You can use SwiftUI APIs to customize how the views integrate with your app.

To use StoreKit views, configure your In-App Purchase metadata in App Store Connect, or in a StoreKit configuration file in Xcode if you're testing your app. Next, create the views using [Store View](#), [ProductView](#), or [SubscriptionStoreView](#). Finally, customize the default views to match your app by using your own icons, backgrounds, and other styling. Use [Previews in Xcode](#) to see your progress as you iterate on your design.

For more information on configuring your In-App Purchase metadata, see [Manage In-App Purchases](#). For more information on StoreKit configuration files in Xcode, see [Setting up StoreKit Testing in Xcode](#).

## Topics

# Merchandising In-App Purchases, subscriptions, and offers

`struct ProductView`

A view that merchandises an individual In-App Purchase product.

`struct StoreView`

A view that merchandises a collection of In-App Purchase products.

`struct SubscriptionStoreView`

A view that merchandises a collection of auto-renewable subscription options that belong to the same subscription group.

`struct SubscriptionOfferView`

`{}` Backyard Birds: Building an app with SwiftData and widgets

Create an app with persistent data, interactive widgets, and an all new in-app purchase experience.

## Styling product views

`nonisolated func productViewStyle(_ style: some ProductViewStyle) -> some View`

Sets the style for In-App Purchase product views within a view.

`nonisolated func productIconBorder() -> some View`

Adds a standard border to an in-app purchase product's icon .

`protocol ProductViewStyle`

A type that specifies the appearance and interaction of In-App Purchase products within the view hierarchy.

`struct ProductViewStyleConfiguration`

The properties of an In-App Purchase product for use by custom product view styles.

## Styling subscription store controls

`nonisolated func subscriptionStoreControlStyle(_ style: some SubscriptionStoreControlStyle) -> some View`

Sets the control style for subscription store views within a view.

```
nonisolated func subscriptionStoreControlStyle<S>(_ style: S, placement : S.Placement) -> some View where S : SubscriptionStoreControlStyle
```

Sets the control style and control placement for subscription store views within a view.

```
protocol SubscriptionStoreControlStyle
```

A type that specifies the appearance and interaction of controls in the subscription store view instances within the view hierarchy.

```
struct SubscriptionStoreControlStyleConfiguration
```

The properties of a subscription store control that includes the list of auto-renewable subscriptions to merchandise.

```
protocol SubscriptionStoreControlPlacement
```

A type that specifies the placement of a subscription control in a subscription store view.

## Styling subscription offer views

```
struct AutomaticSubscriptionOfferViewStyle
```

```
struct CompactSubscriptionOfferViewStyle
```

```
struct SubscriptionOfferViewStyleConfiguration
```

```
protocol SubscriptionOfferViewStyle
```

## Configuring subscription store controls

```
nonisolated func subscriptionStoreControlIcon(@ViewBuilder icon: @escaping (Product, Product.SubscriptionInfo) -> some View) -> some View
```

Sets a view to use to decorate individual subscription options within a subscription store view.

```
nonisolated func subscriptionStorePickerItemBackground(_ background Style: some ShapeStyle) -> some View
```

Sets the background style for picker items of the subscription store view instances within a view.

```
nonisolated func subscriptionStorePickerItemBackground(_ background Style: some ShapeStyle, in shape: some Shape) -> some View
```

Sets the background shape and style for subscription store view picker items within a view.

```
nonisolated func subscriptionStoreButtonLabel(_ label: SubscriptionStoreButtonLabel) -> some View
```

Configures subscription store view instances within a view to use the provided button label.

```
struct SubscriptionStoreButtonLabel
```

The label of the subscribe button that a subscription store view uses.

## Creating custom subscription store control styles

```
struct SubscriptionStoreButton
```

A button for subscribing to an in-app subscription with a localized label and optional caption.

```
struct SubscriptionStorePicker
```

A composite control with a picker for selecting a subscription option and a button for confirming the subscription.

```
struct SubscriptionStorePickerOption
```

A subscription option within a subscription picker control.

## Declaring the structure of a subscription store

```
struct SubscriptionOptionGroup
```

A group of subscription options that includes optional views for labels and marketing content.

```
struct SubscriptionOptionGroupSet
```

A set of groups of subscription options that include optional views for labels and marketing content.

```
struct SubscriptionPeriodGroupSet
```

```
struct SubscriptionOptionSection
```

```
protocol StoreContent
```

A type that represents the content of a store.

```
struct StoreContentBuilder
```

A result builder that creates store content from closures that you provide.

## Styling subscription option groups

```
nonisolated func subscriptionStoreOptionGroupStyle(_ style: some SubscriptionOptionGroupStyle) -> some View
```

Sets the style subscription store views within this view use to display groups of subscription options.

```
func subscriptionStoreOptionGroupStyle(some SubscriptionOptionGroupStyle) -> some StoreContent
```

```
protocol SubscriptionOptionGroupStyle
```

## Adding backgrounds to subscription stores

```
nonisolated func containerBackground<S>(_ style: S, for container: ContainerBackgroundPlacement) -> some View where S : ShapeStyle
```

Sets the container background of the enclosing container using a view.

```
nonisolated func containerBackground<V>(for container: ContainerBackgroundPlacement, alignment: Alignment = .center, @ViewBuilder content: () -> V) -> some View where V : View
```

Sets the container background of the enclosing container using a view.

```
nonisolated func subscriptionStoreControlBackground(_ backgroundStyle: some ShapeStyle) -> some View
```

Set a shape style to use for the background of subscription store view controls within the view.

```
nonisolated func subscriptionStoreControlBackground(_ backgroundStyle: SubscriptionStoreControlBackground) -> some View
```

Set a standard effect to use for the background of subscription store view controls within the view.

```
static var subscriptionStore: ContainerBackgroundPlacement { get }
```

An automatic placement within a subscription store view, based on the view's context.

```
static var subscriptionStoreHeader: ContainerBackgroundPlacement { get }
```

A background placement behind the marketing content of a subscription store view.

```
static var subscriptionStoreFullHeight: ContainerBackgroundPlacement { get }
```

A background placement that spans the full height of a subscription store view.

```
struct SubscriptionStoreControlBackground
```

## Configuring accessory buttons

```
nonisolated func storeButton(_ visibility: Visibility, for buttonKinds: StoreButtonKind...) -> some View
```

Specifies the visibility of auxiliary buttons that store view and subscription store view instances may use.

```
nonisolated func subscriptionStoreSignInAction(_ action: (() -> ()))? -> some View
```

Adds an action to perform when a person uses the sign-in button on a subscription store view within a view.

```
struct StoreButtonKind
```

A button to display in a store view or subscription store view.

```
struct SubscriptionOfferViewButtonKind
```

## Configuring the subscription store policies

```
nonisolated func subscriptionStorePolicyDestination(for button: SubscriptionStorePolicyKind, @ViewBuilder destination: () -> some View) -> some View
```

Configures a view as the destination for a policy button action in subscription store views.

```
nonisolated func subscriptionStorePolicyDestination(url: URL, for button: SubscriptionStorePolicyKind) -> some View
```

Configures a URL as the destination for a policy button action in subscription store views.

```
nonisolated func subscriptionStorePolicyForegroundColorStyle(_ style: some ShapeStyle) -> some View
```

Sets the style for the terms of service and privacy policy buttons within a subscription store view.

```
nonisolated func subscriptionStorePolicyForegroundColorStyle(_ primary: some ShapeStyle, _ secondary: some ShapeStyle) -> some View
```

Sets the primary and secondary style for the terms of service and privacy policy buttons within a subscription store view.

```
struct SubscriptionStorePolicyKind
```

The type of policy, such as the terms of service or privacy policies.

## Selecting subscription offers

```
nonisolated func subscriptionPromotionalOffer(offer: @escaping (Product, Product.SubscriptionInfo) -> Product.SubscriptionOffer?, signature: @escaping (Product, Product.SubscriptionInfo, Product.SubscriptionOffer) async throws -> Product.SubscriptionOffer.Signature) -> some View
```

Selects a promotional offer to apply to a purchase a customer makes from a subscription store view.

Deprecated

```
nonisolated func preferredSubscriptionOffer(_ offer: @escaping (Product, Product.SubscriptionInfo, [Product.SubscriptionOffer]) -> Product.SubscriptionOffer?) -> some View
```

Selects a subscription offer to apply to a purchase that a customer makes from a subscription store view, a store view, or a product view.

```
nonisolated func offerCodeRedemption(isPresented: Binding<Bool>, on Completion: @escaping @MainActor (Result<Void, any Error>) -> Void = { _ in }) -> some View
```

Presents a sheet that enables customers to redeem offer codes that you configure in App Store Connect.

## Configuring purchase options and product descriptions

```
nonisolated func inAppPurchaseOptions(_ options: ((Product) async -> Set<Product.PurchaseOption>?)) -> some View
```

Add a function to call before initiating a purchase from StoreKit view within this view, providing a set of options for the purchase.

```
nonisolated func productDescription(_ visibility: Visibility) -> some View
```

Configure the visibility of labels displaying an in-app purchase product description within the view.

## Responding to store events

```
nonisolated func onInAppPurchaseStart(perform action: ((Product) async -> ()))? -> some View
```

Add an action to perform when a user triggers the purchase button on a StoreKit view within this view.

```
nonisolated func onInAppPurchaseCompletion(perform action: ((Product, Result<Product.PurchaseResult, any Error>) async -> ()))? -> some View
```

Add an action to perform when a purchase initiated from a StoreKit view within this view completes.

## Loading StoreKit data

```
nonisolated func storeProductTask(for id: Product.ID, priority: Task Priority = .medium, action: @escaping (Product.TaskState) async -> ()) -> some View
```

Declares the view as dependent on an In-App Purchase product and returns a modified view.

```
nonisolated func storeProductsTask(for ids: some Collection<String> & Equatable & Sendable, priority: TaskPriority = .medium, action: @escaping (Product.CollectionTaskState) async -> ()) -> some View
```

Declares the view as dependent on a collection of In-App Purchase products and returns a modified view.

```
nonisolated func currentEntitlementTask(for productID: String, priority : TaskPriority = .medium, action: @escaping (EntitlementTaskState< VerificationResult<Transaction>?>) async -> ()) -> some View
```

Declares the view as dependent on the entitlement of an In-App Purchase product, and returns a modified view.

```
nonisolated func subscriptionStatusTask(for groupID: String, priority: TaskPriority = .medium, action: @escaping (EntitlementTaskState<[ Product.SubscriptionInfo.Status]>) async -> ()) -> some View
```

Declares the view as dependent on the status of an auto-renewable subscription group, and returns a modified view.

```
enum CollectionTaskState
```

The state of a task that loads a collection of products in the background.

```
enum TaskState
```

The state of a task that loads a product in the background.

```
enum EntitlementTaskState
```

The state of an entitlement task.

## Requesting a refund

```
@preconcurrency nonisolated func refundRequestSheet(for transactionID: Transaction.ID, isPresented: Binding<Bool>, onDismiss: (@MainActor (Result<Transaction.RefundRequestStatus, Transaction.RefundRequestError>) -> ())? = nil) -> some View
```

Display the refund request sheet for the given transaction.