Article

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.

Overview

With iOS, you can promote In-App Purchases so users can browse them directly on the App Store. Promoted In-App Purchases display on your product page, appear in search results, and may be featured on an appropriate tab on the App Store. Users can start a purchase on the App Store, which takes them into your app to continue the transaction.

If your app isn't installed when the user selects to purchase the in-app product, the App Store automatically downloads the app or prompts the user to purchase it. If the installed version of your app is an older version that doesn't support In-App Purchase promotions, the App Store prompts the user to upgrade the app.

Promoting In-App Purchases requires two steps:

1. In your app, starting in iOS 16.4, implement `PurchaseIntent` to complete the purchase users initiate on the App Store. To support this feature in apps that run in iOS 11–16.3, see Promoting In-App Purchases.

2. In App Store Connect, configure promoted In-App Purchases. For more information, see Promote in-app purchases.

Important

To enable promoted in-app purchases, your app needs to use either `PurchaseIntent` (starting in iOS 16.4) or `paymentQueue(_:shouldAddStorePayment:for:)` (starting in iOS 11). Don't use both at the same time. If necessary, use conditional compilation to identify the OS version the app is running in. For more information, see Running code on a specific platform or OS version.

You can optionally customize which promoted In-App Purchases a users sees on a specific device using the `Product.PromotionInfo` API. Using `Product.PromotionInfo` isn't required for your promoted In-App Purchases to appear on the App Store.

For marketing guidance on this feature, see Promoting Your In-App Purchases.

To test promoted In-App Purchases, see Testing promoted In-App Purchases.

Note

Promoted In-App Purchases aren't available to compatible iPad or iPhone apps running in visionOS.

Set up promoted In-App Purchases using App Store Connect

In App Store Connect, set up promoted In-App Purchases by uploading promotional images. Use the App Store Promotions feature in App Store Connect to manage their default order and visibility. For more information about the setup, see Promote in-app purchases.

Complete the purchase in your app

When users initiate an In-App Purchase on the App Store, StoreKit automatically opens your app and sends the product information in the `intents` asynchronous sequence of `PurchaseIntent`. Your app needs to complete the purchase transaction and perform any related actions that are specific to it.

To complete the transaction, call the `purchase(options:)` method on the `product` of the `PurchaseIntent`. Follow the same workflow your app uses for any In-App Purchase, such as unlocking the purchased content and finishing the transaction (`finish()`).

The following code example listens for purchase intents from the App Store. It receives a `Product` object when the user taps a promoted In-App Purchase on the App Store, and performs its purchase workflow.

```
for await purchaseIntent in PurchaseIntent.intents {
    // Complete the purchase workflow.
    do {
        try await purchaseIntent.product.purchase()
    }
    catch {
        <#Handle Error#>
    }
}
```

Defer or cancel the purchase

In some cases, your app may need to defer or cancel the purchase transaction. For example, you may need to defer the transaction if the user is in the middle of onboarding, and complete it after they finish. Or, you may need to cancel the transaction if the user has already unlocked the product they're trying to buy.

To defer the transaction, store the purchase intent and process it later after the user finishes onboarding or other actions that require deferring.

To cancel the transaction, don't call `purchase(options:)`. Instead, provide feedback to the user. Although this step is optional, if you don't provide feedback, the app's lack of action after the user initiates the purchase on the App Store may seem like a bug.

Customize promoted product order and visibility on a device

To customize promoted In-App Purchases on a device, you can override their default order and visibility using `Product.PromotionInfo`. Use it to show only promotions that are relevant to your user, based on their experience with your app, or based on their previous purchases. For example, suppose a game has In-App Purchases to promote for each game level. You can show only the promoted products for the user's game level, and hide those they already purchased.

To control whether a promoted In-App Purchase appears in the App Store on the device, use any of these methods:

- Add or remove it from the list. Set the product order by calling `updateAll(_:)` or `updateProductOrder(byID:)`. Leave out the products you don't want to display.
- Set its visibility. Show or hide the product by setting its `visibility` value and calling `update()` or `updateAll(_:)`, or call `updateProductVisibility(_:for:)`.

These overrides are specific to the device, and take effect after the user launches the app at least once.

The following code example uses an array of product identifiers to set their order. It calls `currentOrder` to get the `Product.PromotionInfo` objects, and sets their visibility directly. It then calls `updateAll(_:)` to save the changes.

```
// Set the product order.
let orderedProductIdentifiers = [
    "com.example.ExampleApp.product1",
    "com.example.ExampleApp.product2",
    "com.example.ExampleApp.product3"
]

do {
    try await Product.PromotionInfo.updateProductOrder(byID: orderedProductIdentifie
}
catch {
    <#Handle Error#>
}

// Get the current product order.
var promotions: [Product.PromotionInfo] = []

do {
    promotions = try await Product.PromotionInfo.currentOrder
}
catch {
    <#Handle Error#>
}

// Set the visibility for all the products and save the changes.
for i in promotions.indices {
    promotions[i].visibility = .visible
}

do {
    try await Product.PromotionInfo.updateAll(promotions)
}
catch {
    <#Handle Error#>
}
```

The following code example hides a promoted product after the user purchases it:

```
// Change the visibility to hide a promoted product after a user purchases it.
let purchasedProductIdentifier = "com.example.ExampleApp.product1"

do {
    try await Product.PromotionInfo.updateProductVisibility(.hidden, for: purchasedP
}
catch {
    <#Handle Error#>
}
```

Control a promotion’s visibility from App Store Connect

To control a promotion’s visibility from App Store Connect, set its `visibility` value in the app to the default of `Product.PromotionInfo.Visibility.appStoreConnectDefault`. Products with this value are visible or hidden based on the setting you configure in App Store Connect. For more information about using App Store Connect, see Promote in-app purchases.

For example, to promote a product on a holiday for all users, ensure the app sets the product’s visibility to `Product.PromotionInfo.Visibility.appStoreConnectDefault`. In App Store Connect, start with its visibility set as hidden. On the holiday, use App Store Connect to manually change the product to be visible. The promoted product becomes visible in the App Store for all users automatically.

Cancel overrides

To reset changes your app makes to the promoted products’ order and visibility, call the `updateAll(_:)` or `updateProductOrder(byID:)` methods with an empty array. This cancels the overrides so the user can see the promoted In-App Purchases in their default order.

Call `currentOrder` to get a list of your overrides. If the list is empty, the device displays promoted In-App Purchases in their default order.

See Also

Promoted In-App Purchases

`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.