Article

TN3186: Troubleshooting In-App Purchases availability in the sandbox

Identify common configurations that make your In-App Purchases unavailable in the sandbox environment.

Overview

When testing your In-App Purchases in the Apple sandbox environment, your app may not display its products. After submitting your In-App Purchases to App Review, they might also be missing or unresponsive during review. The sandbox environment is a test environment for testing the In-App Purchase implementation in your app with real product data from App Store Connect. TestFlight also uses the sandbox for In-App Purchases. StoreKit queries App Store Connect for this data when your app calls StoreKit APIs in the sandbox. For more information, see Testing In-App Purchases with sandbox.

Important

Testing In-App Purchase in the Apple sandbox environment doesn't require you submit your In-App Purchases for review. For more information, see Explore testing In-App Purchases.

To offer In-App Purchases in your app, call `Product.products(for:)` with a list of product identifiers (`Product ID`) matching these products in the sandbox. `Product.products(for:)` returns an array that includes an instance of `Product` for each of the In-App Purchases. Update your app's UI with these returned instances, which contain all In-App Purchase information configured in App Store Connect for your app.

If `Product.products(for:)` fails to return a `Product` instance for your products in the sandbox, validate your product identifier list and confirm you set a price and added localizations for each of the In-App Purchases.

After you complete the above checks, retry the product request. If the In-App Purchases are still missing, it may be due to the following reasons:

- Your bundle ID doesn't match the bundle ID of an app in App Store Connect.
- Your bundle ID is disabled for the In-App Purchase capability in Certificates, Identifiers & Profiles.
- You are using a wildcard App ID for your app.
- You signed your app with an invalid certificate or provisioning profile.
- You have an inactive Apple Developer Program account.
- You don't have a Paid Apps Agreement in effect in App Store Connect.
- You have incomplete or outdated banking or tax information in App Store Connect.

> Note
>
> If your app fails to display its products when testing In-App Purchases in Xcode, or when launching the app in the App Store, see TN3185: Troubleshooting In-App Purchases availability in Xcode and TN3188: Troubleshooting In-App Purchases availability in the App Store, respectively.

Validate your product identifier list

To verify your product identifier list, perform these steps:

1. In your Xcode project, locate your app's bundle ID.
2. In App Store Connect, find the app that matches your app's bundle ID.
3. Verify each product identifier in your list matches the product identifier (`Product ID`) of an In-App Purchase created for the app in App Store Connect.

Identify the test environment

Apple provides test environments in Xcode and the Apple sandbox to help you verify your implementation of In-App Purchases. For more information, see Testing at all stages of development with Xcode and the sandbox.

If your Xcode project contains a StoreKit Configuration file, ensure you are testing your In-App Purchases in the environment you wish to use. Disable StoreKit Testing in Xcode in your Xcode project to test in the sandbox. Enable StoreKit Testing in Xcode to test in Xcode local test environment.

Use a bundle ID registered in App Store Connect and enabled for In-App Purchase

Testing your In-App Purchases in the sandbox requires that your app uses a bundle ID registered in App Store Connect and enabled for In-App Purchase in Certificates, Identifiers & Profiles. A bundle ID uniquely identifies your app throughout the system, for example, a bundle ID like `com.example` uniquely identifies an app using the bundle ID `com.example.myapp` and you are guaranteed that no other apps are using that same bundle ID. You register a bundle ID for your app when you create an App Store Connect record for the app. App Store Connect displays the registered bundle ID under the App Information section of your app. In your Xcode project, the bundle ID appears in the Signing & Capabilities pane of your app's target. Confirm the bundle ID in your Xcode project matches the bundle ID you have registered for your app.

After you confirm your Xcode project uses a registered bundle ID, verify you have enabled the In-App Purchase capability for the bundle ID. In the Identifiers section of Certificates, Identifiers & Profiles, select your bundle ID, scroll down to In-App Purchase under Capabilities, then confirm you have selected the capability.

> **Note**
>
> In Certificates, Identifiers & Profiles, you can register an explicit App ID or a wildcard App ID for your app. An explicit App ID contains the full path of a bundle ID such as `com.example`. A wildcard App ID contains an asterisk as the last part of its bundle ID search string, for example, `com.example.myapp.*`. For more information, see Register an App ID. The In-App Purchase capability appears enabled by default for an explicit App ID and disabled for a wildcard App ID. In the sandbox, you can't test In-App Purchase with a provisioning profile that contains a wildcard App ID.

Sign your app with a valid certificate and provisioning profile

To use In-App Purchases in your app, you must sign your app with a provisioning profile that grants the app access to the In-App Purchase capability. For more information, see Adding capabilities to your app. A provisioning profile contains App ID and capabilities information. Inspect your provisioning profile, confirm the value of App ID matches your registered bundle ID and capabilities includes In-App Purchase. To view the content of your provisioning profile, see Check Your Provisioning Profile in Diagnosing Issues with Entitlements.

Note

Provisioning profiles that contain a revoked certificate or a modified App ID are invalid and cannot be used to sign your app. If your provisioning profile is invalid or expires, sign your app updates with a regenerated provisioning profile. After you renew your developer account, sign your app updates with new or updated certificates and provisioning profiles. If your development or distribution certificate expires or you revoke it, sign your app updates with a new certificate and updated provisioning profiles. For more information, see Edit a provisioning profile. After you accept and complete an app transfer, sign updates of the transferred app with new provisioning profiles.

Use automatic signing to allow Xcode to manage code signing for you. For more information, see Configure code signing.

Review your Apple Developer Program membership

To build apps with advanced capabilities such as In-App Purchase, you must have an active developer account. For more information, see Choosing a Membership. The following table lists reasons why your account might be inactive:

Reason	Solution
The developer account expired.	Your Account Holder must renew membership, accept the Paid Apps Agreement again, and update tax and banking information. For more information, see Expired memberships.
The most recent version of the Apple Developer Program Licence Agreement is available and unsigned.	Your Account Holder must accept the latest version of the Apple Developer Program Licence Agreement.

Ensure you have a Paid Apps Agreement in effect

To offer In-App Purchases in your app, you must have a Paid Apps Agreement in effect in App Store Connect. The agreement is in effect if App Store Connect shows an `Active` status in its Agreements section. For more information about the statuses, see View agreements status. The following table lists reasons why your Paid Apps Agreement might not be in effect:

Reason	Solution
The Paid Apps Agreement is unsigned.	Your Account Holder must sign the Paid Apps Agreement.
The most recent version of the Paid Apps Agreement is available and unsigned.	Your Account Holder must accept the latest version of the Paid Apps Agreement.
The Paid Apps Agreement expires.	Your Account Holder must renew the Paid Apps Agreement.

Complete all banking and tax information

After you accept the Paid Apps Agreement, your Account Holder needs to submit all required banking and tax information. The banking and tax information are complete if App Store Connect shows an `Active` status in its Bank Accounts and Tax Forms sections, respectively. For more information, see Manage banking information, Manage tax information, and View agreements status.

After you renew the developer membership, check if your Account Holder needs to accept the Paid Apps Agreement and update the financial information again.

Retry your product request later

When you edit In-App Purchase information in App Store Connect, it can take up to 1 hour for your changes to appear in the sandbox environment. For more information, see View and edit in-app purchase information.

Revision History

- **2025-04-29** First published.

See Also

Latest

TN3190: USB audio device design considerations

Learn the best techniques for designing devices that conform to the USB Audio Device Class specifications.

TN3194: Handling account deletions and revoking tokens for Sign in with Apple

Learn the best techniques for managing Sign in with Apple user sessions and responding to account deletion requests.

TN3193: Managing the on-device foundation model’s context window

Learn how to budget for the context window limit of Apple’s on-device foundation model and handle the error when reaching the limit.

TN3115: Bluetooth State Restoration app relaunch rules

Learn about the conditions under which an iOS app will be relaunched by Bluetooth State Restoration.

TN3192: Migrating your iPad app from the deprecated UIRequiresFullScreen key

Support iPad multitasking and dynamic resizing while updating your app to remove the deprecated full-screen compatibility mode.

TN3151: Choosing the right networking API

Learn which networking API is best for you.

TN3111: iOS Wi-Fi API overview

Explore the various Wi-Fi APIs available on iOS and their expected use cases.

TN3191: IMAP extensions supported by Mail for iOS, iPadOS, and visionOS

Learn which extensions to the RFC 3501 IMAP protocol are supported by Mail for iOS, iPadOS, and visionOS.

TN3134: Network Extension provider deployment

Explore the platforms, packaging, OS versions, and device configurations for Network Extension provider deployment.

TN3179: Understanding local network privacy

Learn how local network privacy affects your software.

TN3189: Managing Mail background traffic load

Identify iOS Mail background traffic and manage its impact on your IMAP server.

TN3187: Migrating to the UIKit scene-based life cycle

Update your app to receive scene-based life-cycle events and manage your user interface using scene objects and methods.

- TN3188: Troubleshooting In-App Purchases availability in the App Store
  Verify your In-App Purchases are approved and available for sale in the App Store.
- TN3185: Troubleshooting In-App Purchases availability in Xcode
  Inspect your active StoreKit configuration file for unexpected configurations.
- TN3182: Adding privacy tracking keys to your privacy manifest
  Declare the tracking domains you use in your app or third-party SDK in a privacy manifest.