Article

Running your iOS apps in macOS

Modernize the iOS apps you choose to run on a Mac with Apple silicon, or opt out of running on a Mac altogether.

Overview

iOS Apps on Mac runs your unmodified iPhone and iPad apps on Apple silicon with no porting process. Your apps use the same frameworks and infrastructure that Mac Catalyst apps use to run, but without the need to recompile for the Mac platform.

Although there is no porting process, consider updating your code anyway to provide a better user experience when your app runs in macOS. Adopting modern iOS features in your app makes the transition to macOS simpler because modern features map automatically to appropriate macOS behaviors.

In some cases, you might choose to prevent your app from running in macOS altogether. For example, if you already have a macOS app, or your app relies on features available only on iOS devices, you can choose not to run your app in macOS.

> **Note**
>
> Although you can run your iOS apps unmodified on a Mac with Apple silicon, Mac Catalyst lets you build your app specifically for macOS and customize your app's behavior on that platform. Mac Catalyst also supports deployment on both Apple silicon and Intel-based Mac computers. For more information about how to build your app with Mac Catalyst, see Mac Catalyst.

Determine whether your app makes sense in macOS

iOS and macOS support many of the same frameworks and features, and most iOS apps run smoothly in macOS. However, you might choose to opt out of running your iOS app in macOS under the following circumstances:

- You already created a Mac version of your app using AppKit or Mac Catalyst.
- Your app relies heavily on iOS hardware that's unavailable for Mac, such as accelerometers, gyroscopes, magnetometers, depth-sensing cameras, or GPS.
- Your app requires frameworks, symbols, or features unavailable in macOS.
- Your app's interactions rely extensively on touch input that you can't replicate with the keyboard or other input.
- You don't want users to have access to the content in your app's bundle or data container.
- Your app communicates with custom hardware using the External Accessory framework.

The absence of specific hardware on a Mac doesn't always require you to opt out. You can disable parts of your app that require device-specific hardware, allowing the user to use the rest of your app. Some technologies might also be capable of operating without specific hardware. For example, Core Location doesn't require GPS to return location data. The best approach is to determine what features are available on the current platform using technology-specific APIs, and enable or disable app-specific features accordingly.

You don't need to opt out apps that rely on single-finger touch input or make occasional use of multifinger gestures. The macOS system supports single-finger touch input and standard multifinger gestures for scrolling, zooming, rotating, and pinching using a trackpad. The system also adds a Touch Alternatives tab automatically to your app's Settings window. Touch Alternatives enables keyboard alternatives for tap, swipe, and drag gestures, and enables multifinger gestures using the Option key and a trackpad. However, consider opting out if your app relies extensively on custom, multifinger touch input that has no practical alternative. For example, you might opt out if your app regularly tracks multiple fingers at precise locations on the screen. For more information, see Providing an edge-to-edge, full-screen experience in your iPad app running on a Mac.

If you ultimately decide that it doesn't seem beneficial for your app to run in macOS, you can opt out in App Store Connect, as described below.

Adopt features that offer a better user experience

Some iOS features make it easier for your app to run in macOS and improve the overall user experience. Consider adopting the following features in your iOS apps whenever possible:

- Adopt iPad multitasking and Auto Layout to support resizable windows in macOS. iPad multitasking involves resizing your view controllers to make room for other apps onscreen. If you don't adopt iPad multitasking, your iPad app runs in a fixed-size window. iPhone apps always run in a fixed-size window.
- Optimize your view layouts to improve the experience of live-window resizing. Optimal layouts result in faster animations when the user resizes your window.
- Add iOS keyboard support to your app and create keyboard-based shortcuts for your app's features. Users can access these shortcuts on any iOS or macOS device with a connected keyboard.
- Use `Bundle` and `FileManager` APIs to find files and standard system directories. These APIs always provide you with path locations that are appropriate for the current system.
- Use standard gesture recognizers for pinch, scroll, rotate, and zoom effects. The standard gesture recognizers map automatically to appropriate gestures in macOS.

> **Note**
>
> If you design your iPad app for a full-screen experience (for instance, your app is a game), you can add the keys `UILaunchToFullScreenByDefaultOnMac` and `UISupportsTrueScreenSizeOnMac` to your app's `Info.plist` file to provide a pixel-perfect, edge-to-edge, full-screen experience when the app runs on a Mac.

Audit your code to handle environment differences

When your iOS app runs in macOS, the system automatically maps iOS features to the appropriate replacements in macOS. If you make assumptions about the current environment or device, audit your code to remove those assumptions. If you perform any validation checks, test on a Mac with Apple silicon to make sure those checks continue to work correctly. Specifically, don't assume that:

- Your iOS app's window or other UI elements, including system alerts and pop-ups, are a specific size or at a specific location.
- Multifinger touch input is available.
- iOS-specific hardware is available.

- A front- or rear-facing camera is present. Use `AVCaptureDevice.DiscoverySession` to identify an appropriate camera.
- Files reside at specific file paths.
- Device-specific features are always available.
- The current type of device is always iPhone, iPad, or iPod touch.
- The app window matches the screen resolution.
- PushKit notifications launch your app in the background. In macOS, these notifications launch your app in the foreground instead. Use a notification-service extension to run your code in the background.
- The presence (or absence) of a symbol conveys special meaning. For example, don't assume your app runs on a specific device or platform because of the presence or absence of a symbol.

When you check for specific features, fall back to reasonable default behaviors when those features aren't available. Ensure that the absence of a feature doesn't cause unexpected results, or prevent the user from doing something meaningful with your app. The `isiOSAppOnMac` property of `ProcessInfo` tells you whether your iOS app is running in macOS or iOS, but checking that property should always be your last choice. It's better to run the same code on both platforms.

For more information about how to handle environment differences, see Adapting iOS code to run in the macOS environment.

Test your app in macOS

Xcode supports debugging, testing, and profiling your iOS app natively on a Mac with Apple silicon. When you open your iOS project in Xcode 12 or later, you have the option to build your app and run it directly in macOS. This option doesn't run your app in a Simulator; it runs it as an iOS App on Mac. You can then test whether your app's features work as expected.

App Store features for iOS continue to work when your app runs in macOS, including:

- In-app purchases and subscriptions
- App capabilities and entitlements
- On-demand resources
- App thinning

When you use app thinning to optimize your app for different devices and operating systems, the App Store selects the resources and content that offer the best fit for a Mac. It then removes any other resources to create a streamlined installation of your app. When you export your app from Xcode 12 or later, you can test the thinning support using the Mac virtual thinning target.

Use TestFlight to distribute your app to your testers, or create an archive and export it using the ad-hoc or development distribution method. During the export process, Xcode creates an appropriately signed app for you to distribute to your testers. For more information, see Distributing your app to registered devices.

Choose whether to include your iOS app in the Mac App Store

After you sign the updated developer agreement, the App Store automatically makes compatible iOS apps available to users of a Mac with Apple silicon. However, if you're planning to ship a macOS version of your app, or if your app doesn't make sense for a Mac, you can change your app's availability in App Store Connect.

1. Open the App Store Connect API page in App Store Connect.

2. Disable the "Make this app available on Mac" option.

After you remove your app's availability for Mac, the Mac App Store stops offering your app for sale. Users who previously downloaded the app for Mac can still use it, but can't download it again.

See Also

iOS apps on Mac

- Adapting iOS code to run in the macOS environment

  Support modern iOS features that result in a better user experience when running on Apple silicon.

- {} Providing touch gesture equivalents using Touch Alternatives

  Enable Touch Alternatives to provide keyboard, mouse, and trackpad equivalents to your iOS app when it runs on a Mac with Apple silicon.

- {} Providing an edge-to-edge, full-screen experience in your iPad app running on a Mac

  Take advantage of the true native resolution of a Mac display when running your iPad app in full-screen mode on a Mac.