Article

Determining whether to bring your app to visionOS

Decide whether to bring your existing iPadOS or iOS app to visionOS.

Overview

If you have an existing app that runs in iPadOS or iOS and you want to bring it to visionOS, you can choose to either run your compatible app as-is or create a visionOS-specific version.

visionOS supports most of the same technologies as iOS, so many apps built to run on iPad or iPhone can run unmodified on Apple Vision Pro. When a compatible app runs in visionOS, it retains the same appearance it has in iPadOS or iOS, and its content appears in a window in the person's surroundings. Apps built specifically for visionOS adopt the standard system appearance, look more natural on the platform, and support 3D content and immersive experiences.

The visionOS SDK has many differences from the iOS SDK. Frameworks and features might behave differently or be unavailable when your app runs in visionOS. Use this article to ensure your app behaves as you expect on every platform.

Decide whether your app suits the platform

In some cases, you may decide your app is not well-suited to visionOS due to platform differences. For example, consider the following types of apps:

- **Apps that act as containers for app extensions.** This includes apps where the primary purpose is to deliver custom keyboard extensions, device drivers, sticker packs, SMS and MMS message-filtering extensions, call directory extensions, or widgets.
- **Navigation-based apps.** This includes apps that follow a person's location changes, such as apps that offer turn-by-turn directions or navigation.

- **Selfie or camera-based apps.** This includes apps where the primary purpose is to capture images or video from the device's cameras. The device cameras are unavailable, but continuity camera and persona capture are available on Apple Vision Pro.

Opt out of running on visionOS

If you don't want your app to run on Apple Vision Pro, change your app's availability in App Store Connect:

1. Select your app in App Store Connect.
2. Navigate to the Pricing and Availability information.
3. Disable the "Make this app available on Apple Vision Pro" option.

When you remove your app's availability for Apple Vision Pro, the App Store stops making your iOS app available for visionOS. People who already downloaded your app on their Vision Pro can still run it in visionOS, but they can't download it again and won't receive any updates. This setting doesn't affect the version of your app built natively using the visionOS SDK.

For information on managing your app's availability for Apple Vision Pro, see Manage availability of iPhone and iPad apps on Apple Vision Pro.

Run as a compatible app on visionOS

visionOS runs compatible iPad and iPhone apps by linking against the iOS SDK to provide continuous access to existing content right away. visionOS supports most of the same technologies as iOS, so many apps built to run on iPad or iPhone can run unmodified on Apple Vision Pro. Read Making your existing app compatible with visionOS to learn how to make your app run successfully as a compatible app in visionOS.

Create a visionOS version of your app

Apps built specifically for visionOS adopt the standard system appearance, and they look more natural on the platform. Creating a version of your app specifically for visionOS gives you the opportunity to add elements that work well on the platform, such as 3D content and immersive experiences. Read Bringing your existing apps to visionOS for more information on how to modify your app to run successfully as a visionOS app and how to update your interface to take advantage of visionOS-specific elements.

Check framework availability in visionOS

The following frameworks are deprecated in their entirety in both iOS and visionOS. If your app still uses these frameworks, stop using them immediately. The reference documentation for each framework includes information about how to update your code.

- Accounts
- Address Book
- Address Book UI
- Assets Library
- GLKit
- iAd
- Newsstand Kit
- Notification Center

> **Important**
>
> visionOS removed many deprecated symbols entirely, turning these deprecation warnings into missing-symbol errors on the platform.

Here's a list of frameworks and features that behave differently in visionOS:

- **ActivityKit.** Available on iOS only. Check the `areActivitiesEnabled` property of `ActivityAuthorizationInfo` to determine if Live Activities are authorized.
- **AirPlay.** visionOS hides AirPlay sharing buttons in system interfaces, and you can't use AirPlay features from compatible apps.
- **App extensions.** visionOS doesn't load App Clips, device drivers, device activity monitors, keyboard extensions, Messages app extensions, photo-editing app extensions, SMS and call-reporting extensions, or widgets.
- **Apple Watch features.** visionOS ignores watchOS apps and WatchKit extensions in your iOS or iPadOS app. Face sharing in ClockKit does nothing in visionOS.
- **ARKit.** This framework requires you to use different APIs for iOS and visionOS, as visionOS can't display windows that contain ARKit views. Check the `isSupported` property of your configuration object to determine availability of augmented reality features. In visionOS, ARKit views such as `ARView` are never available, so isolate interface code containing those views to the iOS version of your app. For information about how to bring an ARKit app to visionOS, see Bringing your ARKit app to visionOS.

- **Audio and video.** visionOS doesn't support Picture in Picture or AV routing features. Check the availability of video features before using them. Be prepared for audio playback to stop automatically when your app moves to the background.
- **Automatic Assessment Configuration.** Check for error values when you configure an `AEAssessmentSession` object. The framework returns an error if you try to start a test in visionOS.
- **AVFoundation.** Identify what cameras are available using the `AVCaptureDevice.DiscoverySession` class. Don't assume the presence of specific cameras. Capture interfaces aren't available in visionOS. Use availability checks to determine which services are present.
- **Cellular telephony and CallKit.** You can still implement Voice-over-IP (VoIP) services using CallKit and Core Telephony. Phone number verification, call-blocking, and other cellular-related services are unavailable.
- **Contacts.** Use the `CNContactStore` class to determine your app's authorization status.
- **Core Bluetooth.** Use the `CBCentralManager` and `CBPeripheralManager` classes to determine feature availability and your app's authorization status.
- **Core Haptics.** visionOS plays audio feedback instead of haptic feedback. Call the `capabilitiesForHardware()` method of the haptic engine to determine the available features.
- **Core Location.** You can request location using the standard location service, but most other services are unavailable. Use availability checks to determine which services are present. The Always authorization level is unavailable and automatically becomes When in Use authorization. Check the properties of `CLLocationManager` to determine the availability of location services.
- **Core Motion.** Barometer data is unavailable, but most other sensors are available. Use availability checks to determine which sensors you can use. Check the properties of `CMMotionManager` to determine the availability of accelerometers, gyroscopes, magnetometers, and other hardware sensors.
- **Core NFC.** Check the `readingAvailable` property of your reader session to determine if NFC tag reading is available.
- **Device management.** Calls to the ManagedSettings and ManagedSettingsUI frameworks do nothing in visionOS.
- **EventKit.** Use the `EKEventStore` class to determine your app's authorization status.
- **Exposure Notification.** Use the `ENManager` class to determine your app's authorization status.
- **Game controllers.** visionOS delivers game controller events only when someone is looking at your app. To require a game controller as an input device, add the `GCRequiresControllerUserInteraction` key with the visionOS value to your app's `Info.plist`.

- **Handoff.** visionOS doesn't attempt to hand off user activities to other devices.
- **HomeKit.** You can't add accessories using a QR code from Apple Vision Pro. Check the properties of `HMHomeManager` to determine your app's authorization status.
- **Local Authentication.** Use the `LAContext` class to determine the authentication policies you can use.
- **MapKit.** User-tracking features that involve heading information aren't available in visionOS.
- **Media Player.** Some APIs are unavailable in visionOS. Use the `MPMediaLibrary` class to determine your app's authorization status.
- **Metrics.** You can use MetricKit to gather on-device diagnostic logs and generate reports, but you can't gather metrics in visionOS.
- **Multi-Touch.** The system reports a maximum of two simultaneous touch inputs — one for each of the person's hands. All system gesture recognizers handle these inputs appropriately, including for zoom and rotation gestures that require multiple fingers. If you have custom gesture recognizers that require more than two points of interaction, update them to support only one or two touches in visionOS.
- **MusicKit.** Some APIs are unavailable in visionOS.
- **Nearby Interaction.** The framework does nothing in visionOS. Check the `deviceCapabilities` property of your session to determine whether features are available.
- **Parental controls.** Calls to the FamilyControls framework do nothing in visionOS.
- **PencilKit.** visionOS doesn't report touches of type `UITouch.TouchType.pencil`, but it does report other types of touches.
- **PhotoKit.** Use the `PHPhotoLibrary` class to determine your app's authorization status.
- **ProximityReader.** Check the `isSupported` property of the card reader object to determine if Tap to Pay on iPhone is available.
- **Push to Talk.** These services are unavailable in visionOS. Check for errors when creating a `PTChannelManager`.
- **ReplayKit.** Check the `isAvailable` property of `RPScreenRecorder` to determine if screen recording support is available.
- **RoomPlan.** Check the `isSupported` property of the `RoomCaptureSession` object to determine if LiDAR scanning is available on the device.
- **Safari Services.** A link that presents a `SFSafariViewController` now opens a new scene in the Safari app.
- **Screen Time.** Calls to the Screen Time framework do nothing in visionOS.

- **Sensor-related features.** The SensorKit framework is unavailable in the native visionOS SDK and calls to it do nothing when running a compatible app. Use the `SRSensorReader` class to determine your app's authorization status for compatible apps.
- **Social media.** Calls to the Social framework do nothing in visionOS.
- **Speech.** Use the `SFSpeechRecognizer` class to determine if speech recognition is available.
- **System interfaces.** Authorization prompts, Sign in with Apple prompts, and other system-provided interfaces run asynchronously outside of your app's process. Because these interfaces don't run modally in your app, your app might not receive immediate responses.
- **User Notifications.** Use the `getNotificationSettings(completionHandler:)` method of `UNUserNotificationCenter` to determine your app's authorization status.
- **Vehicle features.** The system doesn't call your app's CarPlay code. Calls you make using CarKey do nothing in visionOS.
- **VisionKit.** The `DataScannerViewController` APIs are unavailable, but other features are still available. Data scanners do nothing in VisionKit in visionOS.
- **Watch Connectivity.** The framework supports connections only between an iPhone and Apple Watch. Call the `isSupported()` method of the `WCSession` object to determine if the framework is available.

> **Note**
>
> Review Building spatial experiences for business apps with enterprise APIs for visionOS for information on enterprise APIs and how to request the entitlements.

See Also

iOS migration and compatibility

- Bringing your existing apps to visionOS

  Build a version of your iPadOS or iOS app using the visionOS SDK, and update your code for platform differences.

- Bringing your ARKit app to visionOS

  Update an iPadOS or iOS app that uses ARKit, and provide an equivalent experience in visionOS.

- Making your existing app compatible with visionOS

Modify your iPadOS or iOS app to run successfully in visionOS as a compatible app.