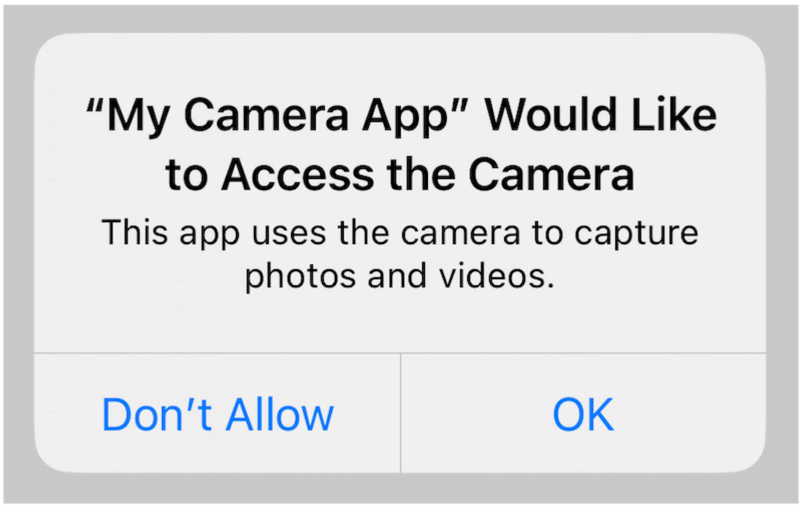
Article

Requesting authorization to capture and save media

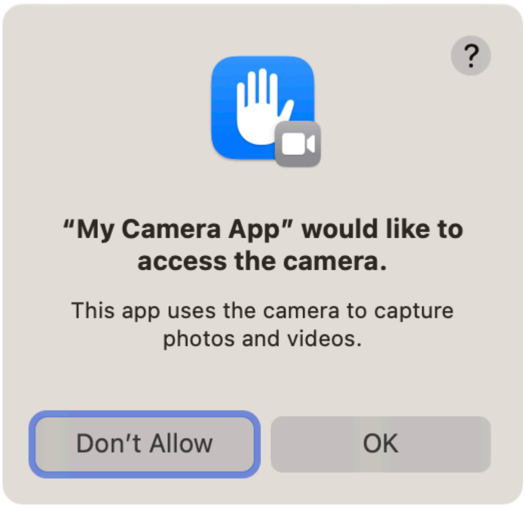
Prompt the user to authorize access to the camera, microphone, and photo library.

Overview

In iOS and macOS 10.14 and later, the user must explicitly grant permission for each app to access the camera and microphone. Before your app can use a capture device for the first time, the system presents an alert with an app-specific message that you specify, to ask the user to grant your app access to the capture device. For example, the following screenshots show the camera access alert in iOS and macOS 10.14 and later:


"My Camera App" Would Like to Access the Camera

This app uses the camera to capture photos and videos.



“My Camera App” would like to access the camera.

This app uses the camera to capture photos and videos.

The system remembers the user’s response to each access alert, so subsequent uses of the corresponding capture device don’t cause the alert to appear again. The user can change permission settings for your app in Settings > Privacy in iOS and in System Preferences > Security & Privacy in macOS.

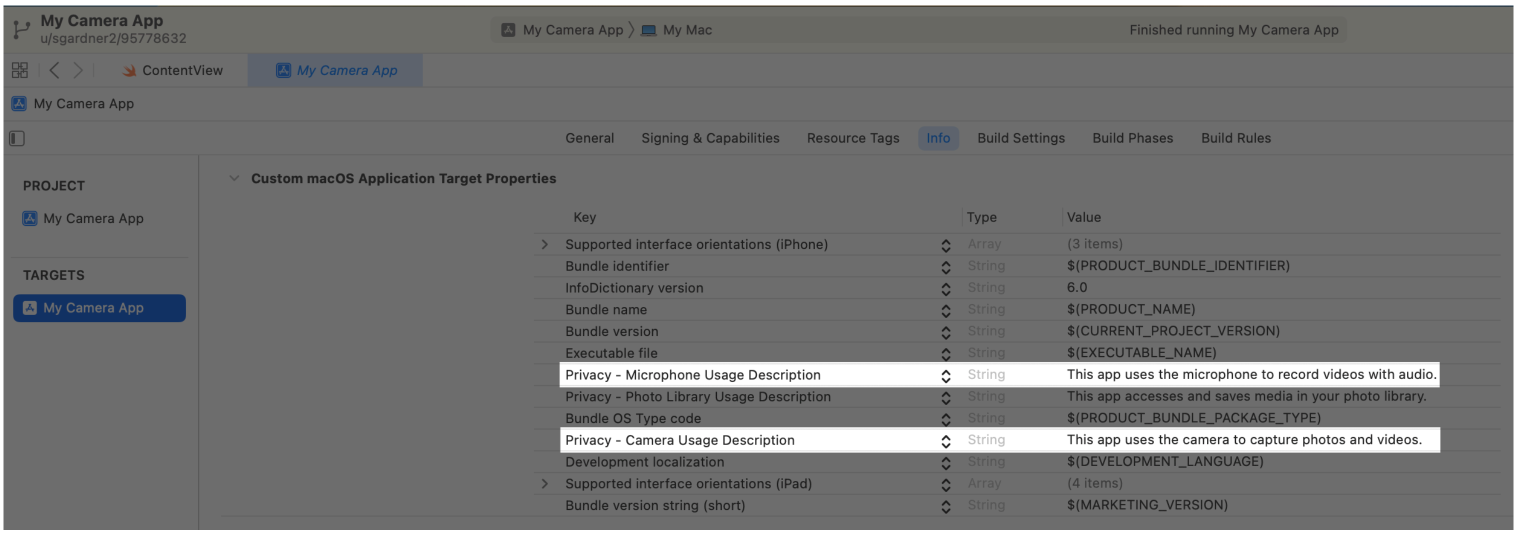
This article outlines the steps you can take to ensure your app has permission before it attempts to capture media.

Configure access alerts

The system requires that your app provides static messages to display to the user when the system asks for permission to access the camera or microphone:

- If your app uses device cameras, include the `NSCameraUsageDescription` key in your app’s `Info.plist` file.
- If your app uses device microphones, include the `NSMicrophoneUsageDescription` key in your app’s `Info.plist` file.

For each key, provide a message that explains to the user why your app needs to capture media so the user can feel confident granting permission to your app.



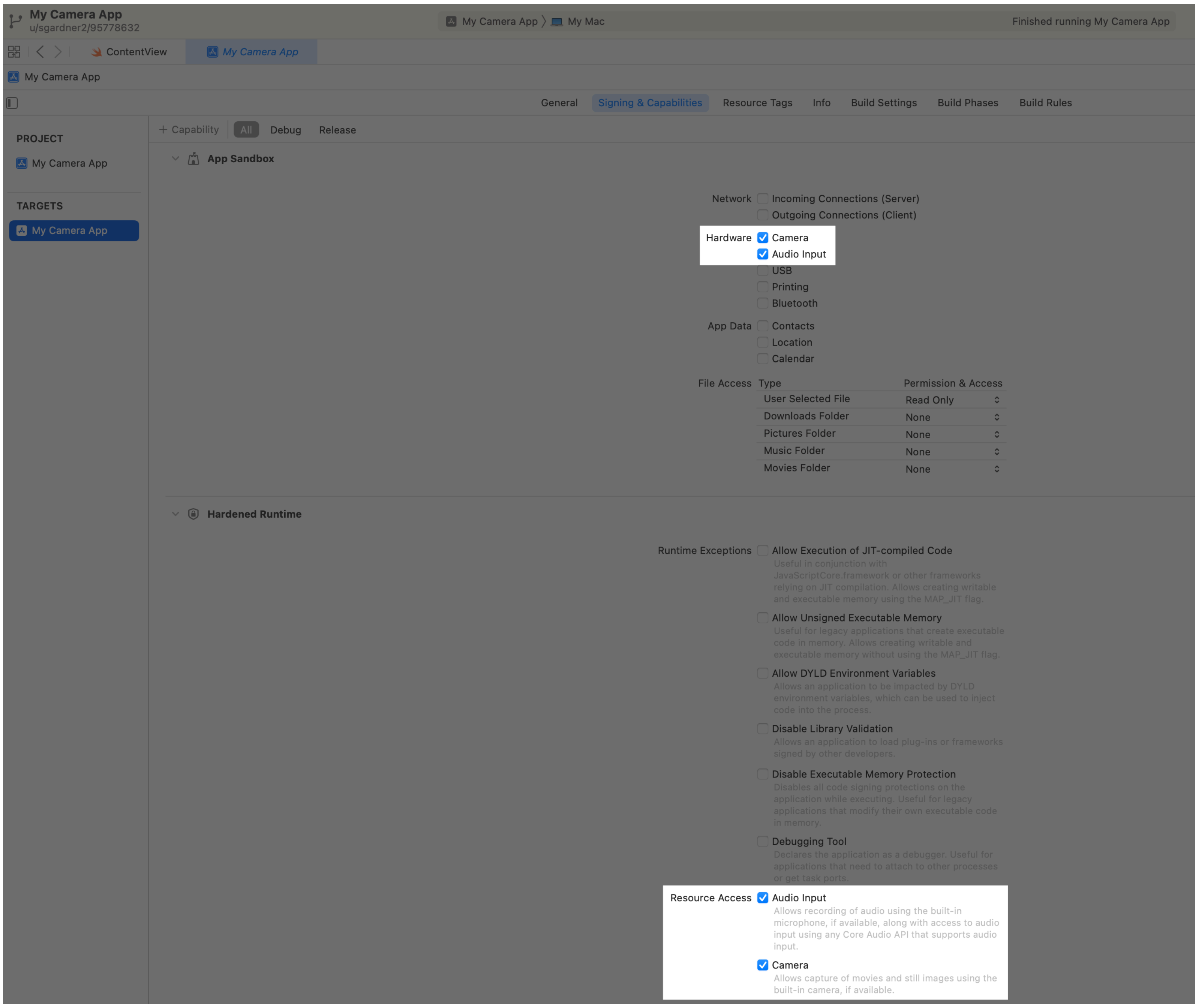
Custom macOS Application Target Properties

Key	Type	Value
Supported interface orientations (iPhone)	Array	(3 items)
Bundle identifier	String	$(PRODUCT_BUNDLE_IDENTIFIER)
InfoDictionary version	String	6.0
Bundle name	String	$(PRODUCT_NAME)
Bundle version	String	$(CURRENT_PROJECT_VERSION)
Executable file	String	$(EXECUTABLE_NAME)
Privacy - Microphone Usage Description	String	This app uses the microphone to record videos with audio.
Privacy - Photo Library Usage Description	String	This app accesses and saves media in your photo library.
Bundle OS Type code	String	$(PRODUCT_BUNDLE_PACKAGE_TYPE)
Privacy - Camera Usage Description	String	This app uses the camera to capture photos and videos.
Development localization	String	$(DEVELOPMENT_LANGUAGE)
Supported interface orientations (iPad)	Array	(4 items)
Bundle version string (short)	String	$(MARKETING_VERSION)

Enable entitlements in macOS

In macOS, you also need to enable the following entitlements in Signing & Capabilities for your app targets:

- If your app uses device cameras, enable the `Camera entitlement`.
- If your app uses device microphones, enable the `Audio Input Entitlement`.



My Camera App
u/sgardner2/95778632
My Camera App 〉 My Mac
Finished running My Camera App
ContentView | My Camera App
My Camera App

General | Signing & Capabilities | Resource Tags | Info | Build Settings | Build Phases | Build Rules

PROJECT
My Camera App
TARGETS
My Camera App

+ Capability | All | Debug | Release

App Sandbox

Network: Incoming Connections (Server); Outgoing Connections (Client)
Hardware: [✓ Camera] [✓ Audio Input] USB; Printing; Bluetooth
App Data: Contacts; Location; Calendar

File Access

Type	Permission & Access
User Selected File	Read Only
Downloads Folder	None
Pictures Folder	None
Music Folder	None
Movies Folder	None

Hardened Runtime

Runtime Exceptions:
- Allow Execution of JIT-compiled Code
  Useful in conjunction with JavaScriptCore.framework or other frameworks relying on JIT compilation. Allows creating writable and executable memory using the MAP_JIT flag.
- Allow Unsigned Executable Memory
  Useful for legacy applications that create executable code in memory. Allows creating writable and executable memory without using the MAP_JIT flag.
- Allow DYLD Environment Variables
  Allows an application to be impacted by DYLD environment variables, which can be used to inject code into the process.
- Disable Library Validation
  Allows an application to load plug-ins or frameworks signed by other developers.
- Disable Executable Memory Protection
  Disables all code signing protections on the application while executing. Useful for legacy applications that modify their own executable code in memory.
- Debugging Tool
  Declares the application as a debugger. Useful for applications that need to attach to other processes or get task ports.

Resource Access:
- [✓ Audio Input]
  Allows recording of audio using the built-in microphone, if available, along with access to audio input using any Core Audio API that supports audio input.
- [✓ Camera]
  Allows capture of movies and still images using the built-in camera, if available.

Important

Your app needs to contain the appropriate key in its `Info.plist` file, and the appropriate entitlement enabled in macOS, before it requests authorization or attempts to use a capture device. Otherwise, the system terminates your app.

Verify and request authorization for capture

Always test the AVCaptureDevice authorizationStatus(for:) method before setting up a capture session. If the user hasn't granted or denied capture permission, the authorization status is AVAuthorizationStatus.notDetermined. In this case, use the requestAccess(for:completionHandler:) method to tell the system to prompt the user.

Call requestAccess(for:completionHandler:) before starting capture, but only at a time that's appropriate for your app. For example, if photo or video capture isn't the main focus of your app, check for permission only when the user invokes your app's camera-related features. Here's an example:

```
var isAuthorized: Bool {
    get async {
        let status = AVCaptureDevice.authorizationStatus(for: .video)

        // Determine if the user previously authorized camera access.
        var isAuthorized = status == .authorized

        // If the system hasn't determined the user's authorization status,
        // explicitly prompt them for approval.
        if status == .notDetermined {
            isAuthorized = await AVCaptureDevice.requestAccess(for: .video)
        }

        return isAuthorized
    }
}

func setUpCaptureSession() async {
    guard await isAuthorized else { return }
    // Set up the capture session.
}
```

Request authorization to save captured media

Your app must get permission to access the photo library before it attempts to save photos or videos; otherwise the system terminates your app. Which permission you request depends on the types of media you want to save:

- For most photo and video capture workflows, including Live Photos and RAW format capture, use the PHPhotoLibrary and PHAssetCreationRequest classes. These classes require read/write access to the photo library, so you need to use the NSPhotoLibraryUsage

`Description` key in your `Info.plist` file to provide the user a message to request access. For additional information, see Delivering an Enhanced Privacy Experience in Your Photos App.

- If your app only needs to save video to the photo library, the `UISaveVideoAtPathToSavedPhotosAlbum(_:_:_:_:)` function is a simpler alternative to `PHPhotoLibrary`. This function only requires write access to the library. Use the `NSPhotoLibraryAddUsageDescription` key in your `Info.plist` file to provide the user a message to request permission to save to the photo library.

Note

The `UIImageWriteToSavedPhotosAlbum(_:_:_:_:)` function isn't recommended for use with photos captured with `AVCapturePhotoOutput` because `UIImage` doesn't support the features and metadata included in photo output.