Article

Adopting Picture in Picture in a Standard Player

Add Picture in Picture (PiP) playback to your app using a player view controller.

Overview

The `AVPlayerViewController` provides the standard video playback experience across iOS, iPadOS, and tvOS. In tvOS, it supports a wide variety of remotes, skipping, scanning, scrubbing, Siri commands, interstitial support, and more. After you configure your audio session and set the project capabilities as described in Configuring your app for media playback, your player automatically supports PiP playback. When your app runs on a supported device, the user can manage PiP in the standard player.

Familiarize Yourself with the PiP Controls

PiP playback starts when the user selects the PiP button in the player interface. In iOS and iPadOS, PiP playback starts automatically if your video is playing in full-screen mode and the user exits the app. When a video isn't filling the entire screen in width, use `canStartPictureInPictureAutomaticallyFromInline` to indicate a video is the primary focus. In either case, the player window minimizes to a movable, floating window. In general, the system automatically pauses the video upon scene backgrounding, so you don't need to pause video based on activation state.

Tip

In iOS and iPadOS, you can disable automatic invocation of Picture in Picture in Settings > General > Picture in Picture. Check this setting if you think you've set up everything correctly but find that your video doesn't enter PiP mode when you return to the Home screen.

Selecting the stop button in the PiP interface terminates PiP and restores video playback within your app. AVKit can't make assumptions about how you designed your app, so it doesn't know how to properly restore your video playback interface. Instead, it delegates that responsibility to you.

Starting in iOS 14, the PiP user interface provides controls that allow users to skip forward and backward within a video. The system enables these controls by default for apps linked in iOS 14 or later. If you need to restrict skipping content for legal disclaimers or advertisements, use `requiresLinearPlayback` during the required section of your video. Set this property back to `false` once you can allow seeking again.

Restore Your Video Playback Interface

To handle the restore process, your code must adopt the `AVPlayerViewControllerDelegate` protocol and implement the `playerViewController(_:restoreUserInterfaceForPictureInPictureStopWithCompletionHandler:)` method. The framework calls this method when control returns to your app, giving you the opportunity to determine how to properly restore your video player's interface. If you originally presented your video player using the `present(_:animated:completion:)` method of `UIViewController`, restore your player interface in the same way in the delegate callback method.

```
func playerViewController(_ playerViewController: AVPlayerViewController,
                          restoreUserInterfaceForPictureInPictureStopWithCompletionH
    present(playerViewController, animated: false) {
        completionHandler(true)
    }
}
```

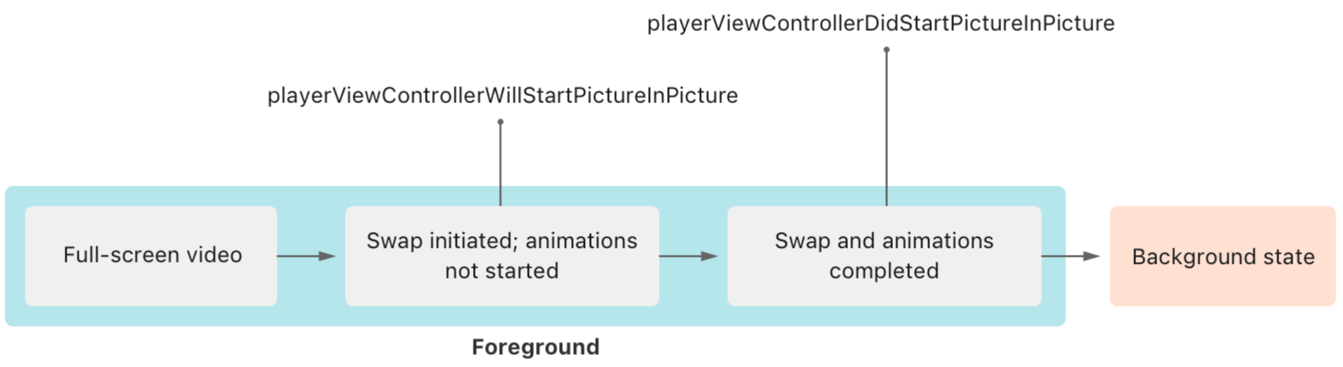
Avoid adding animations during the swap so you can ensure quick restoration for your user.

Important

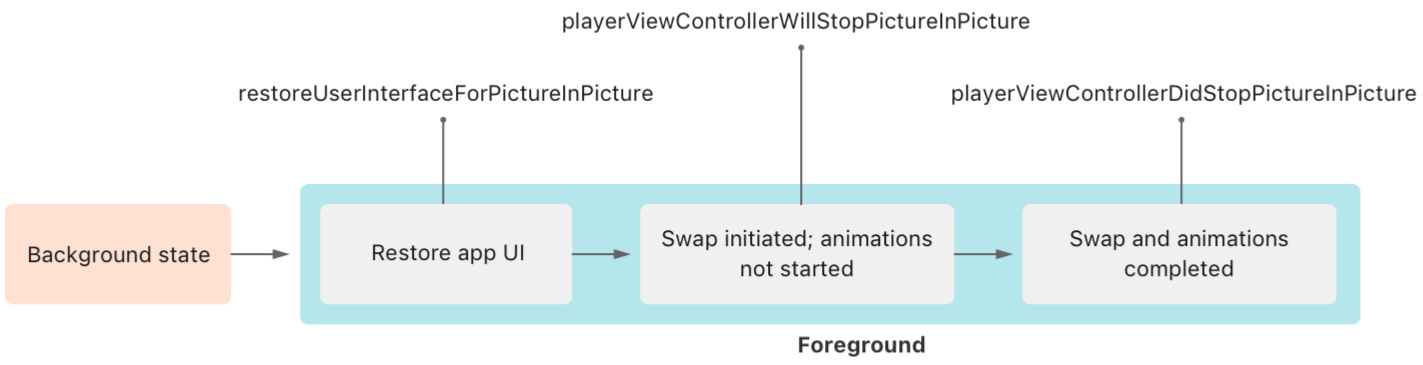
To allow the system to finish restoring your user interface, call the completion handler with a value of `true`.

Swap PiP Content in tvOS

In tvOS, users can play videos in PiP alongside a full-screen video. Video playback can move between PiP and full screen, so your app needs to be ready to handle the swap to provide a seamless PiP experience. When you swap your content with another app, consider your content that's going into PiP and your content leaving PiP to go full screen. The illustration below shows the life cycle of your full-screen content swapping with another app's PiP.

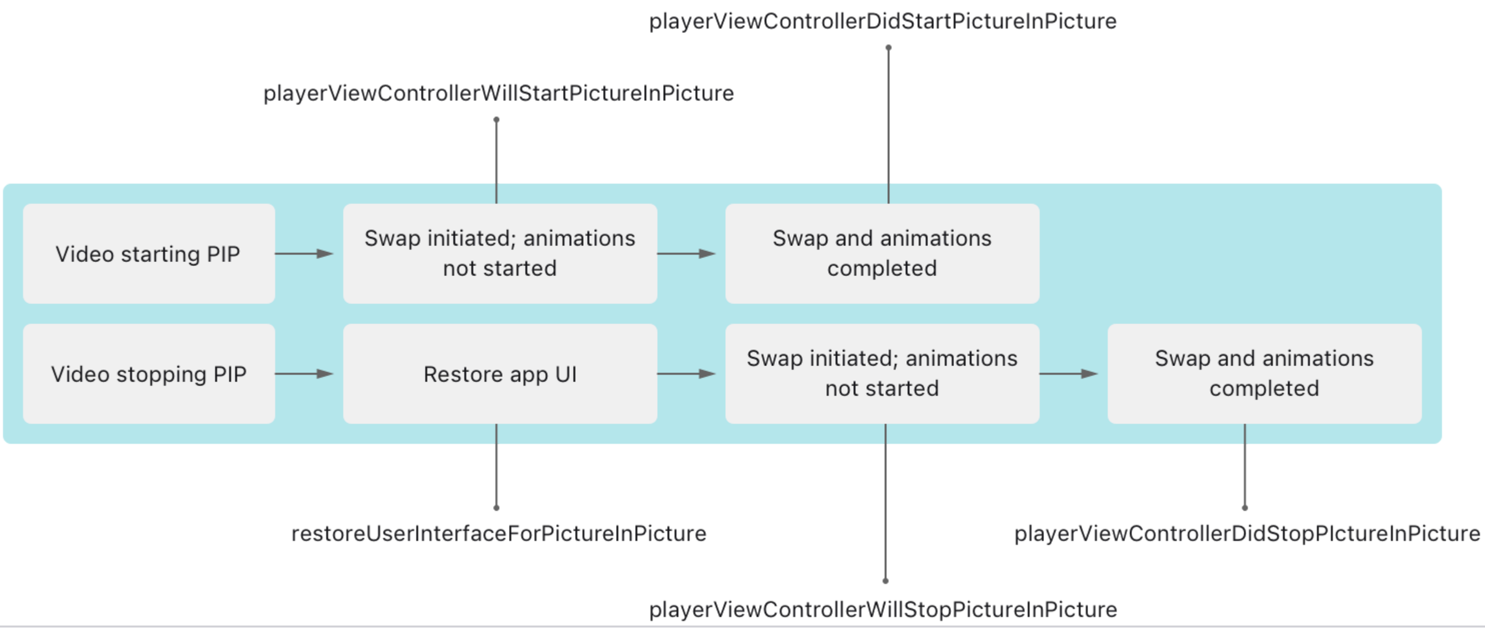

playerViewControllerDidStartPictureInPicture
playerViewControllerWillStartPictureInPicture
Full-screen video
Swap initiated; animations not started
Swap and animations completed
Background state
Foreground

The illustration below shows the life cycle of your app's PiP content moving to full screen.


playerViewControllerWillStopPictureInPicture
restoreUserInterfaceForPictureInPicture
playerViewControllerDidStopPictureInPicture
Background state
Restore app UI
Swap initiated; animations not started
Swap and animations completed
Foreground

When you swap content within your app, you need to handle both sides of the life cycle events from both video players. The following sequence shows you the callbacks to expect with a video that swaps content between PiP and full screen.

1. The video starting PiP receives `playerViewControllerWillStartPictureInPicture(_:)`, but the system hasn't started the animations.
2. The video going full screen receives `playerViewController(_:restoreUserInterfaceForPictureInPictureStopWithCompletionHandler:)` to restore its full-screen user interface.
3. The video going full screen receives `playerViewControllerWillStopPictureInPicture(_:)`, and the system hasn't started the animations.
4. The video starting PiP receives `playerViewControllerDidStartPictureInPicture(_:)`, and the system completes the swap and animations for the video that started PiP.
5. The video going full screen receives `playerViewControllerDidStopPictureInPicture(_:)`, and the system completes the swap and animations for the video that moved full screen.



playerViewControllerDidStartPictureInPicture
playerViewControllerWillStartPictureInPicture
Video starting PIP
Swap initiated; animations not started
Swap and animations completed
Video stopping PIP
Restore app UI
Swap initiated; animations not started
Swap and animations completed
restoreUserInterfaceForPictureInPicture
playerViewControllerDidStopPIctureInPicture
playerViewControllerWillStopPictureInPicture

See Also

Picture in Picture

{} Adopting Picture in Picture Playback in tvOS

Add advanced multitasking capabilities to your video apps by using Picture in Picture playback in tvOS.

Adopting Picture in Picture in a Custom Player

Add controls to your custom player user interface to invoke Picture in Picture (PiP) playback.

Adopting Picture in Picture for video calls

Add multitasking capability to your video-call apps by using Picture in Picture (PiP).

Accessing the camera while multitasking on iPad

Operate the camera in Split View, Slide Over, Picture in Picture, and Stage Manager modes.

`class AVPictureInPictureController`

A controller that responds to user-initiated Picture in Picture playback of video in a floating, resizable window.