Class

AVPlayerViewController

A view controller that displays content from a player and presents a native user interface to control playback.

iOS 8.0+ | iPadOS 8.0+ | Mac Catalyst 13.1+ | tvOS 9.0+ | visionOS 1.0+

```
@MainActor
class AVPlayerViewController
```

Mentioned in

- Adopting the system player interface in visionOS
- Customizing the tvOS Playback Experience
- Adopting Picture in Picture in a Custom Player
- Trimming and exporting media in visionOS
- Working with Interstitial Content

Overview

A player view controller makes it simple to add media playback capabilities to your app that match the styling and features of the native system players. Using this object also means that your app automatically adopts the new features and styling of future operating system releases.

Important

The framework doesn't support subclassing `AVPlayerViewController`.

Support AirPlay

AirPlay lets users stream media to Apple TV, HomePod, and AirPlay 2-compatible speakers and smart TVs. A player view controller supports AirPlay automatically, but you need to configure your app to enable it. See Configuring your app for media playback for more information about configuring your app for background playback.

Adopt Picture in Picture playback

`AVPlayerViewController` provides Picture in Picture (PiP) playback in iOS, tvOS, and macOS. PiP playback lets users minimize the video player to a small floating window so they can perform other activities in the primary app or in another app.

Customize the tvOS playback experience

`AVPlayerViewController` in tvOS brings advanced Siri Remote control features to your app. This support lets users play and navigate your content, and access supporting features like subtitles and alternate audio tracks. This object also provides support for using Siri Remote voice commands, such as “Skip ahead 15 seconds” and “What did they say?”, to control playback of your content.

AVKit for tvOS extends the features of `AVPlayerViewController` and `AVPlayerItem` to provide additional ways to navigate and present content. Features unique to the tvOS player user interface include:

- Navigation Marker Groups. A group of navigation markers that allow a viewer to jump between significant events in the media timeline. The most common type of navigation marker group is a chapter list, but you can also create additional or alternative means of navigation — for example, to allow the user to quickly jump between key moments in a recorded sporting event. The player view controller lets the user choose between multiple marker groups for navigating through the media timeline.

Use the `AVNavigationMarkersGroup` class to create and describe navigation markers, then use the `navigationMarkerGroups` property to associate marker groups with the current `AVPlayerItem` object.

- Interstitial Content. Some content might not relate to the main content that your app presents, or might have different presentation requirements. For example, you might not allow the user to skip over advertisements when scrubbing through the playback timeline, or to skip mandatory legal notices.

Use the `AVInterstitialTimeRange` class to describe interstitial content, and the `interstitialTimeRanges` property to associate those time ranges with the current `AVPlayerItem` object.

- Content Proposals. When presenting serialized content, like a TV show, you often want to propose additional content for the viewer to watch when the current episode ends. It’s straightforward to add this functionality to your app using content proposals.

Use the `AVContentProposal` class to describe the proposed content, and set it as the `nextContentProposal` property of the current `AVPlayerItem` object. You can implement the methods of the player view controller’s `delegate` object to prepare to present a content proposal, and perform actions in response to the viewer accepting, rejecting, or deferring the proposal.

Topics

Configuring presentation

`var showsPlaybackControls: Bool`

A Boolean value that indicates whether the player view controller shows playback controls.

`var contentOverlayView: UIView?`

A view that displays between the video content and the playback controls.

`var videoGravity: AVLayerVideoGravity`

A string that specifies how the video displays within the bounds of the view controller’s view.

`var videoBounds: CGRect`

The size and position of the video image within the bounds of the view controller’s view.

`var showsTimecodes: Bool`

A Boolean value that determines whether the player view displays timecodes, if available.

`var appliesPreferredDisplayCriteriaAutomatically: Bool`

A Boolean value that indicates whether the view controller automatically sets the screen’s display criteria to match that of the currently playing asset.

Customizing the tvOS player UI

`var playbackControlsIncludeTransportBar: Bool`

A Boolean value that indicates whether the player shows the transport bar and related controls.

`var playbackControlsIncludeInfoViews: Bool`

A Boolean value that indicates whether the player presents video metadata, navigation markers, and playback settings views when the user requests them.

`var transportBarIncludesTitleView: Bool`

A Boolean value that indicates whether the player user interface shows the title view above the scrubber.

`var transportBarCustomMenuItems: [UIMenuElement]`

An array of actions and menus to display with the default player controls.

`var customInfoViewControllers: [UIViewController]`

An array of view controllers to display as content tabs in the player user interface.

`var infoViewActions: [UIAction]!`

An array of actions to present in the Info content view.

`var contextualActions: [UIAction]`

An array of action controls to present contextually during playback.

`var customOverlayViewController: UIViewController?`

A view controller that presents custom content over the player view.

`var unobscuredContentGuide: UILayoutGuide`

A layout guide that represents an area that fixed-position playback controls don't obscure when visible.

~~`var customInfoViewController: UIViewController?`~~

A view controller that provides client-specific content and controls alongside system-provided information and settings panels.

Deprecated

Configuring the visionOS player UI

`var infoViewActions: [UIAction]!`

An array of actions to present in the Info content view.

`var customInfoViewControllers: [UIViewController]`

An array of view controllers to display as content tabs in the player user interface.

`var contextualActions: [UIAction]`

An array of action controls to present contextually during playback.

`var contextualActionsInfoView: UIView`

A view the system shows adjacent to the contextual actions that's suitable for showing related information.

`var contextualActionsPreviewImage: UIImage?`

An image to show alongside the contextual actions.

`var requiresMonoscopicViewingMode: Bool`

A Boolean value that indicates whether to permit playback of 2D video content only.

`var experienceController: AVExperienceController`

The experience controller for this view controller.

`var groupExperienceCoordinator: AVGroupExperienceCoordinator`

The group experience coordinator for this view controller.

Presenting the visionOS trimming UI

`var canBeginTrimming: Bool`

A Boolean value that indicates whether the current media supports trimming.

`func beginTrimming(completionHandler: ((Bool) -> Void)?)`

Presents the system trimming interface controls inside the player view.

Configuring frame analysis

`var allowsVideoFrameAnalysis: Bool`

A Boolean value that indicates whether to perform video frame analysis.

`var toggleLookupAction: UIAction`

An action that enables the visual lookup interface.

`var videoFrameAnalysisTypes: AVVideoFrameAnalysisType`

The types of analysis a player view controller performs on a paused video frame.

`struct AVVideoFrameAnalysisType`

Constants that define the types of analysis a player view controller may perform on a paused video frame.

Configuring playback speed

`var speeds: [AVPlaybackSpeed]`

A list of user-selectable playback speeds to show in the playback speed control.

`var selectedSpeed: AVPlaybackSpeed?`

The currently selected playback speed.

`func selectSpeed(AVPlaybackSpeed)`

Selects a specified playback speed.

`class AVPlaybackSpeed`

An object that represents a user-selectable playback speed in a playback user interface.

Configuring Picture in Picture

`var allowsPictureInPicturePlayback: Bool`

A Boolean value that indicates whether the player allows Picture in Picture playback.

`var canStartPictureInPictureAutomaticallyFromInline: Bool`

A Boolean value that indicates whether Picture in Picture starts automatically when transitioning to the background when the view controller presents its content inline.

Managing full-screen behavior

`var entersFullScreenWhenPlaybackBegins: Bool`

A Boolean value that determines whether the player automatically displays in full screen when the user taps the play button.

`var exitsFullScreenWhenPlaybackEnds: Bool`

A Boolean value that indicates whether the player exits full-screen mode when playback ends.

Managing subtitles

`var allowedSubtitleOptionLanguages: [String]?`

An array of language codes that restrict the set of subtitle languages available to the user.

`var requiresFullSubtitles: Bool`

A Boolean value that indicates whether the user can disable the display of subtitles.

Preventing navigation

`var requiresLinearPlayback: Bool`

A Boolean value that determines whether the player allows the user to skip media content.

Configuring skipping behavior

`var isSkipForwardEnabled: Bool`

A Boolean value that indicates whether forward-skipping is available.

`var isSkipBackwardEnabled: Bool`

A Boolean value that indicates whether backward-skipping is available.

`var skippingBehavior: AVPlayerViewControllerSkippingBehavior`

The behavior that skipping gestures perform.

`enum AVPlayerViewControllerSkippingBehavior`

Constants that represent the player view controller’s skipping behavior.

Determining display readiness

`var isReadyForDisplay: Bool`

A Boolean value that indicates whether the player item’s first video frame is ready for display.

Updating Now Playing information

`var updatesNowPlayingInfoCenter: Bool`

A Boolean value that indicates whether the view controller updates Now Playing information.

Proposing additional content

`var contentProposalViewController: AVContentProposalViewController!`

The view controller responsible for the presentation of content proposals.

Accessing the player

`var player: AVPlayer?`

The player object that provides the media content for the view controller to display.

Accessing the delegate object

`var delegate: (any AVPlayerViewControllerDelegate)?`

The delegate object for the player view controller.

Configuring pixel buffers

`var pixelBufferAttributes: [String : Any]?`

The pixel buffer attributes of the video frames the view controller presents.

High dynamic range

`var preferredDisplayDynamicRange: AVDisplayDynamicRange`

Describes how High Dynamic Range (HDR) video content renders.

`enum AVDisplayDynamicRange`

Describes how High Dynamic Range (HDR) video content renders.

Type Properties

`class var mediaCharacteristicsForSupportedCustomMediaSelectionSchemes: [AVMediaCharacteristic]`

Relationships

Inherits From

`UIViewController`

Conforms To

`CVarArg`
`CustomDebugStringConvertible`
`CustomStringConvertible`
`Equatable`
`Hashable`
`NSCoding`
`NSExtensionRequestHandling`
`NSObjectProtocol`

`NSTouchBarProvider`
`Sendable`
`SendableMetatype`
`UIActivityItemsConfigurationProviding`
`UIAppearanceContainer`
`UIContentContainer`
`UIFocusEnvironment`
`UIPasteConfigurationSupporting`
`UIResponderStandardEditActions`
`UIStateRestoring`
`UITraitChangeObservable`
`UITraitEnvironment`
`UIUserActivityRestoring`

See Also

iOS playback and capture

{} Playing video content in a standard user interface

Play media full screen, embedded inline, or in a floating Picture in Picture (PiP) window using a player view controller.

`protocol AVPlayerViewControllerDelegate`

A protocol that defines the methods to implement to respond to player view controller events.

`class AVCaptureEventInteraction`

An object that registers handlers to respond to capture events from system hardware buttons.

`class AVCaptureEvent`

An object that describes a user interaction with a system hardware button.

`class AVCaptureEventSound`

A sound object for a capture event.

`class AVInputPickerInteraction`

Use `AVInputPickerInteraction` to present an input picker.