Class

AVPlayer

An object that provides the interface to control the player's transport behavior.

iOS 4.0+ | iPadOS 4.0+ | Mac Catalyst 13.1+ | macOS 10.7+ | tvOS 9.0+ | visionOS 1.0+ | watchOS 1.0+

```
@MainActor
class AVPlayer
```

Mentioned in

- Controlling the transport behavior of a player
- Observing playback state in SwiftUI
- Supporting AirPlay in your app
- Implementing simple enhanced buffering for your content
- Monitoring playback progress in your app

Overview

A player is a controller object that manages the playback and timing of a media asset. Use an instance of `AVPlayer` to play local and remote file-based media, such as QuickTime movies and MP3 audio files, as well as audiovisual media served using HTTP Live Streaming.

Use a player object to play a single media asset. You can reuse the player instance to play additional media assets using its `replaceCurrentItem(with:)` method, but it manages the playback of only a single media asset at a time. The framework also provides a subclass called `AVQueuePlayer` that you can use to manage the playback of a queue of media assets.

You use an `AVPlayer` to play media assets, which AVFoundation represents using the `AVAsset` class. `AVAsset` only models the *static* aspects of the media, such as its duration or creation date,

and on its own, isn't suitable for playback with an `AVPlayer`. To play an asset, you create an instance of its *dynamic* counterpart found in `AVPlayerItem`. This object models the timing and presentation state of an asset played by an instance of `AVPlayer`. See the `AVPlayerItem` reference for more details.

`AVPlayer` is a dynamic object whose state continuously changes. There are two approaches you can use to observe a player's state:

- **General State Observations:** You can use key-value observing (KVO) to observe state changes to many of the player's dynamic properties, such as its `currentItem` or its playback `rate`.
- **Timed State Observations:** KVO works well for general state observations, but isn't intended for observing continuously changing state like the player's time. `AVPlayer` provides two methods to observe time changes:
- `addPeriodicTimeObserver(forInterval:queue:using:)`
- `addBoundaryTimeObserver(forTimes:queue:using:)`

These methods let you observe time changes either periodically or by boundary, respectively. As changes occur, invoke the callback block or closure you supply to these methods to give you the opportunity to take some action such as updating the state of your player's user interface.

`AVPlayer` and `AVPlayerItem` are nonvisual objects, meaning that on their own they're unable to present an asset's video onscreen. There are two primary approaches you use to present your video content onscreen:

- **AVKit:** The best way to present your video content is with the AVKit framework's `AVPlayerViewController` class in iOS and tvOS, or the `AVPlayerView` class in macOS. These classes present the video content, along with playback controls and other media features giving you a full-featured playback experience.
- **AVPlayerLayer:** When building a custom interface for your player, use `AVPlayerLayer`. You can set this layer a view's backing layer or add it directly to the layer hierarchy. Unlike `AVPlayerView` and `AVPlayerViewController`, a player layer doesn't present any playback controls—it only presents the visual content onscreen. It's up to you to build the playback transport controls to play, pause, and seek through the media.

Alongside the visual content presented with AVKit or `AVPlayerLayer`, you can also present animated content synchronized with the player's timing using `AVSynchronizedLayer`. Use a synchronized layer pass along player timing to its layer subtree. You can use `AVSynchronizedLayer` to build custom effects in Core Animation, such as animated lower thirds or video transitions, and have them play in sync with the timing of the player's current `AVPlayerItem`.

Topics

Creating a player

`init(url: URL)`

Creates a new player to play a single audiovisual resource referenced by a given URL.

`init(playerItem: AVPlayerItem?)`

Creates a new player to play the specified player item.

`init()`

Creates a player object.

Managing the player item

`var currentItem: AVPlayerItem?`

The item for which the player is currently controlling playback.

`func replaceCurrentItem(with: AVPlayerItem?)`

Replaces the current item with a new item.

Determining player readiness

`var status: AVPlayer.Status`

A value that indicates the readiness of a player object for playback.

`enum Status`

Status values that indicate whether a player can successfully play media.

`var error: (any Error)?`

An error that caused a failure.

Controlling playback

`var defaultRate: Float`

A default rate at which to begin playback.

`func play()`

Begins playback of the current item.

`func pause()`

Pauses playback of the current item.

```
var rate: Float
```

The current playback rate.

```
class let rateDidChangeNotification: NSNotification.Name
```

A notification that a player posts when its rate changes.

Observing playback time

```
func currentTime() -> CMTime
```

Returns the current time of the current player item.

```
func addPeriodicTimeObserver(forInterval: CMTime, queue: dispatch_queue_t?, using: (CMTime) -> Void) -> Any
```

Requests the periodic invocation of a given block during playback to report changing time.

```
func addBoundaryTimeObserver(forTimes: [NSValue], queue: dispatch_queue_t?, using: () -> Void) -> Any
```

Requests the invocation of a block when specified times are traversed during normal playback.

```
func removeTimeObserver(Any)
```

Cancels a previously registered periodic or boundary time observer.

Seeking through media

```
func seek(to: CMTime)
```

Requests that the player seek to a specified time.

```
func seek(to: CMTime, completionHandler: (Bool) -> Void)
```

Requests that the player seek to a specified time, and to notify you when the seek is complete.

```
func seek(to: CMTime, toleranceBefore: CMTime, toleranceAfter: CMTime)
```

Requests that the player seek to a specified time with the amount of accuracy specified by the time tolerance values.

```
func seek(to: CMTime, toleranceBefore: CMTime, toleranceAfter: CMTime, completionHandler: (Bool) -> Void)
```

Requests that the player seek to a specified time with the amount of accuracy specified by the time tolerance values, and to notify you when the seek is complete.

`func seek(to: Date)`

Requests that the player seek to a specified date.

`func seek(to: Date, completionHandler: (Bool) -> Void)`

Requests that the player seek to a specified date, and to notify you when the seek is complete.

Configuring waiting behavior

`var automaticallyWaitsToMinimizeStalling: Bool`

A Boolean value that indicates whether the player should automatically delay playback in order to minimize stalling.

`var reasonForWaitingToPlay: AVPlayer.WaitingReason?`

The reason the player is currently waiting for playback to begin or resume.

`struct WaitingReason`

The reasons a player is waiting to begin or resume playback.

`var timeControlStatus: AVPlayer.TimeControlStatus`

A value that indicates whether playback is in progress, paused indefinitely, or waiting for network conditions to improve.

`enum TimeControlStatus`

Constants that indicate the state of playback control.

`func playImmediately(atRate: Float)`

Plays the available media data immediately, at the specified rate.

Responding when playback ends

`var actionAtItemEnd: AVPlayer.ActionAtItemEnd`

The action to perform when the current player item has finished playing.

`enum ActionAtItemEnd`

The actions a player can take when it finishes playing.

Configuring media selection criteria

`var appliesMediaSelectionCriteriaAutomatically: Bool`

A Boolean value that indicates whether the receiver should apply the current selection criteria automatically to player items.

`func mediaSelectionCriteria(forMediaCharacteristic: AVMediaCharacteristic) -> AVPlayerMediaSelectionCriteria?`

Returns the automatic selection criteria for media items with the specified media characteristic.

`func setMediaSelectionCriteria(AVPlayerMediaSelectionCriteria?, forMediaCharacteristic: AVMediaCharacteristic)`

Applies automatic selection criteria for media that has the specified media characteristic.

Accessing player output

`var videoOutput: AVPlayerVideoOutput?`

The video output for this player.

Configuring audio behavior

`var volume: Float`

The audio playback volume for the player.

`var isMuted: Bool`

A Boolean value that indicates whether the audio output of the player is muted.

`var allowedAudioSpatializationFormats: AVAudioSpatializationFormats`

The source audio channel layouts the player item supports for spatialization.

~~`var isAudioSpatializationAllowed: Bool`~~

A Boolean value that indicates whether the player item allows spatialized audio playback.

Deprecated

`var audioOutputSuppressedDueToNonMixableAudioRoute: Bool`

Whether the player's audio output is suppressed due to being on a non-mixable audio route.

`var intendedSpatialAudioExperience: any SpatialAudioExperience`

The player's intended Spatial Audio experience.

Configuring background playback

`var audiovisualBackgroundPlaybackPolicy: AVPlayerAudiovisualBackgroundPlaybackPolicy`

A policy that determines how playback of audiovisual media continues when the app transitions to the background.

`enum AVPlayerAudiovisualBackgroundPlaybackPolicy`

Policies that describe playback behavior when an app transitions to the background while playing video.

Managing external playback

`var allowsExternalPlayback: Bool`

A Boolean value that indicates whether the player allows switching to external playback mode.

`var isExternalPlaybackActive: Bool`

A Boolean value that indicates whether the player is currently playing video in external playback mode.

`var usesExternalPlaybackWhileExternalScreenIsActive: Bool`

A Boolean value that indicates whether the player should automatically switch to external playback mode while the external screen mode is active.

`var externalPlaybackVideoGravity: AVLayerVideoGravity`

The video gravity of the player for external playback mode only.

Determining HDR playback eligibility

`class var eligibleForHDRPlayback: Bool`

A Boolean value that indicates whether the current device can present content to an HDR display.

~~`class var availableHDRModes: AVPlayer.HDRMode`~~

The HDR modes that are available for playback.

Deprecated

~~`struct HDRMode`~~

A bitfield type that specifies an HDR mode.

Deprecated

`class let eligibleForHDRPlaybackDidChangeNotification: NSNotification.Name`

A notification that's posted whenever HDR playback eligibility changes.

Coordinating playback

`var playbackCoordinator: AVPlayerPlaybackCoordinator`

The playback coordinator for the player.

Synchronizing multiple players

`func setRate(Float, time: CMTime, atHostTime: CMTime)`

Synchronizes the playback rate and time of the current item with an external source.

`func preroll(atRate: Float, completionHandler: ((Bool) -> Void)?)`

Begins loading media data to prime the media pipelines for playback.

`func cancelPendingPrerolls()`

Cancels any pending preroll requests and invokes the corresponding completion handlers, if present.

`var sourceClock: CMClock?`

A clock the player uses for item time bases.

~~`var masterClock: CMClock?`~~

The host clock for item time bases.

Deprecated

Preventing sleep and backgrounding

`var preventsDisplaySleepDuringVideoPlayback: Bool`

A Boolean value that indicates whether video playback prevents display and device sleep.

`var preventsAutomaticBackgroundingDuringVideoPlayback: Bool`

A Boolean value that indicates whether video playback prevents the system from automatically backgrounding the app.

Determining content protections

`var isOutputObscuredDueToInsufficientExternalProtection: Bool`

A Boolean value that indicates whether output is being obscured because of insufficient external protection.

Configuring audio and video devices

`var audioOutputDeviceUniqueID: String?`

Specifies the unique ID of the Core Audio output device used to play audio.

`var preferredVideoDecoderGPURegistryID: UInt64`

The registry identifier for the GPU used for video decoding.

Configuring the network resource priority

`var networkResourcePriority: AVPlayer.NetworkResourcePriority`

Indicates the priority of this player for network bandwidth resource distribution.

`enum NetworkResourcePriority`

This defines the network resource priority for a player.

Configuring observation

`class var isObservationEnabled: Bool`

AVPlayer and other AVFoundation types can optionally be observed using Swift Observation.

Configuring AirPlay behavior

~~`var allowsAirPlayVideo: Bool`~~

A Boolean value that indicates whether the player allows AirPlay video playback.

Deprecated

~~`var isAirPlayVideoActive: Bool`~~

A Boolean value that indicates whether the player is playing video through AirPlay.

Deprecated

~~`var usesAirPlayVideoWhileAirPlayScreenIsActive: Bool`~~

A Boolean value that indicates whether the player automatically switches to AirPlay Video while AirPlay Screen is active.

Deprecated

Displaying closed captions

~~`var isClosedCaptionDisplayEnabled: Bool`~~

A Boolean value that indicates whether the player uses closed captioning.

Deprecated

Relationships

Inherits From

`NSObject`

Inherited By

`AVQueuePlayer`

Conforms To

`AVRoutingPlaybackParticipant`
`CVarArg`
`Copyable`
`CustomDebugStringConvertible`
`CustomStringConvertible`
`Equatable`
`Hashable`
`NSObjectProtocol`
`Observable`
`Sendable`

See Also

Playback control

Observing playback state in SwiftUI

Keep your user interface in sync with state changes from playback objects.

Controlling the transport behavior of a player

Play, pause, and seek through a media presentation.

{} Creating a seamless multiview playback experience

Build advanced multiview playback experiences with the AVFoundation and AVRouting frameworks.

`class AVPlayerItem`

An object that models the timing and presentation state of an asset during playback.

`class AVPlayerItemTrack`

An object that represents the presentation state of an asset track during playback.

`class AVQueuePlayer`

An object that plays a sequence of player items.

`class AVPlayerLooper`

An object that loops media content using a queue player.