Class

# AVPlayerItem

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

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

```
@MainActor
class AVPlayerItem
```

## Mentioned in

- Controlling the transport behavior of a player
- Implementing simple enhanced buffering for your content
- Observing playback state in SwiftUI
- Selecting subtitles and alternative audio tracks

## Overview

A player item stores a reference to an `AVAsset` object, which represents the media to play. If you require inspecting an asset before you enqueue it for playback, call its `load(_:isolation:)` method to retrieve the values of one or more properties. Alternatively, you can tell the player item to automatically load the required properties by passing them to its `init(asset:automaticallyLoadedAssetKeys:)` initializer. When the player item is ready to play, the asset properties you request are ready to use.

## Topics

## Creating a player item

`convenience init(url: URL)`

Creates a player item with a specified URL.

`convenience init(asset: AVAsset)`

Creates a player item for a specified asset.

`convenience init(asset: any AVAsset & Sendable)`

`convenience init(asset: AVAsset, automaticallyLoadedAssetKeys: [AVPartialAsyncProperty<AVAsset>])`

Creates a player item for the asset, and automatically loads values for the specified properties.

`convenience init(asset: any AVAsset & Sendable, automaticallyLoadedAssetKeys: [AVPartialAsyncProperty<AVAsset>])`

`init(asset: AVAsset, automaticallyLoadedAssetKeys: [String]?)`

Creates a player item with the specified asset and the asset keys to automatically load.

## Accessing tracks

`var tracks: [AVPlayerItemTrack]`

An array of player item track objects.

## Accessing metadata

`var externalMetadata: [AVMetadataItem]`

An array of additional metadata for the player item to supplement or replace an asset's embedded metadata.

## Determining readiness

`var status: AVPlayerItem.Status`

The status of the player item.

`enum Status`

The statuses for a player item.

`var error: (any Error)?`

The error that caused the player item to fail.

## Determining playback capabilities

`var canPlayReverse: Bool`

A Boolean value that indicates whether the item can play in reverse.

`var canPlayFastForward: Bool`

A Boolean value that indicates whether the item can be fast forwarded.

`var canPlayFastReverse: Bool`

A Boolean value that indicates whether the item can be quickly reversed.

`var canPlaySlowForward: Bool`

A Boolean value that indicates whether the item can play slower than normal.

`var canPlaySlowReverse: Bool`

A Boolean value that indicates whether the item can play slowly backward.

## Setting playback boundaries

`var forwardPlaybackEndTime: CMTime`

The time at which forward playback ends.

`var reversePlaybackEndTime: CMTime`

The time at which reverse playback ends.

## Stepping through media

`var canStepForward: Bool`

A Boolean value that indicates whether the item supports stepping forward.

`var canStepBackward: Bool`

A Boolean value that indicates whether the item supports stepping backward.

`func step(byCount: Int)`

Moves the player item's current time forward or backward by a specified number of steps.

## Seeking through media

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

Sets the current playback time to the specified time.

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

Sets the current playback time within a specified time bound and invokes the specified block when the seek operation completes or is interrupted.

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

Sets the current playback time to the time specified by the date object.

`func cancelPendingSeeks()`

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

## Selecting media options

`func select(AVMediaPresentationSetting, for: AVMediaSelectionGroup)`

When the associated AVPlayer's appliesMediaSelectionCriteriaAutomatically property is set to YES, configures the player item to prefer a particular presentation setting, replacing any previous preference for settings of the same media presentation selector.

`var preferredCustomMediaSelectionSchemes: [AVCustomMediaSelectionScheme]`

Indicates the AVCustomMediaSelectionSchemes of AVMediaSelectionGroups of the receiver's asset with which an associated UI implementation should configure its interface for media selection.

`func effectiveMediaPresentationSettings(for: AVMediaSelectionGroup) -> [AVMediaPresentationSelector : Any]`

Indicates the media presentation settings with media characteristics that are possessed by the currently selected AVMediaSelectionOption in the specified AVMediaSelectionGroup.

`func selectMediaPresentationLanguage(String, for: AVMediaSelectionGroup)`

When the associated AVPlayer's appliesMediaSelectionCriteriaAutomatically property is set to YES, configures the player item to prefer a particular language, replacing any previous preference for available languages of the specified group's custom media selection scheme.

`func selectedMediaPresentationLanguage(for: AVMediaSelectionGroup) -> String?`

Returns the selected media presentation language for the specified media selection group, if any language has previously been selected via use of -selectMediaPresentationLanguages:forMediaSelectionGroup:.

```
func selectedMediaPresentationSettings(for: AVMediaSelectionGroup) -> [AVMediaPresentationSelector : Any]
```

Indicates the media presentation settings that have most recently been selected for each AVMediaPresentationSelector of the AVCustomMediaSelectionScheme of the specified AVMediaSelectionGroup.

```
var currentMediaSelection: AVMediaSelection
```

The current media selections for each of the receiver's media selection groups.

```
func select(AVMediaSelectionOption?, in: AVMediaSelectionGroup)
```

Selects a media option in a given media selection group and deselects all other options in that group.

```
func selectMediaOptionAutomatically(in: AVMediaSelectionGroup)
```

Selects the media option in the specified media selection group that best matches the receiver's automatic selection criteria.

## Setting variant behavior

```
var variantPreferences: AVVariantPreferences
```

The preferences the player item uses when selecting variant playlists.

```
struct AVVariantPreferences
```

Defines the preferences the player item uses when selecting variant playlists.

```
var startsOnFirstEligibleVariant: Bool
```

A Boolean value that indicates whether playback starts with the first eligible variant that appears in the stream's main playlist.

## Configuring interstitial events

```
var integratedTimeline: AVPlayerItemIntegratedTimeline
```

An integrated timeline that represents the player item timing including its scheduled interstitial events.

```
var automaticallyHandlesInterstitialEvents: Bool
```

A Boolean value that indicates whether the player item automatically plays interstitial events according to server-side directives.

`var translatesPlayerInterstitialEvents: Bool`

A Boolean value that indicates whether the player translates interstitial events to interstitial time ranges.

`var interstitialTimeRanges: [AVInterstitialTimeRange]`

An array of time ranges that identify interstitial content.

`var template: AVPlayerItem?`

The template player item that initializes this instance.

## Accessing timing information

`func currentTime() -> CMTime`

Returns the current time of the item.

`func currentDate() -> Date?`

Returns the current time of the item as a date.

`var duration: CMTime`

The duration of the item.

`var timebase: CMTimebase?`

The timebase information for the item.

## Determining available time ranges

`var loadedTimeRanges: [NSValue]`

An array of time ranges indicating media data that is readily available.

`var seekableTimeRanges: [NSValue]`

An array of time ranges within which it is possible to seek.

## Determining buffering status

`var isPlaybackLikelyToKeepUp: Bool`

A Boolean value that indicates whether the item will likely play through without stalling.

`var isPlaybackBufferFull: Bool`

A Boolean value that indicates whether the internal media buffer is full and that further I/O is suspended.

`var isPlaybackBufferEmpty: Bool`

A Boolean value that indicates whether playback has consumed all buffered media and that playback will stall or end.

## Configuring expensive network behavior

`var preferredPeakBitRateForExpensiveNetworks: Double`

A limit of network bandwidth consumption by the item when connecting over expensive networks.

`var preferredMaximumResolutionForExpensiveNetworks: CGSize`

An upper limit on the resolution of video to download when connecting over expensive networks.

## Accessing text style rules

`var textStyleRules: [AVTextStyleRule]?`

An array of text style rules that specify the formatting and presentation of Web Video Text Tracks (WebVTT) subtitles.

`class AVTextStyleRule`

An object that represents the text styling rules to apply to a media item's textual content.

## Accessing logging information

`func accessLog() -> AVPlayerItemAccessLog?`

Returns an object that represents a snapshot of the network access log.

`class AVPlayerItemAccessLog`

An object used to retrieve the access log associated with a player item.

`class AVPlayerItemAccessLogEvent`

A single entry in a player item's access log.

`func errorLog() -> AVPlayerItemErrorLog?`

Returns an object that represents a snapshot of the error log.

`class AVPlayerItemErrorLog`

The error log associated with a player item.

`class` `AVPlayerItemErrorLogEvent`

A single item in a player item's error log.

## Observing notifications

`class let didPlayToEndTimeNotification: NSNotification.Name`

A notification the system posts when a player item plays to its end time.

`class let failedToPlayToEndTimeNotification: NSNotification.Name`

A notification that the system posts when a player item fails to play to its end time.

`class let timeJumpedNotification: NSNotification.Name`

A notification the system posts when a player item's time changes discontinuously.

`class let playbackStalledNotification: NSNotification.Name`

A notification the system posts when a player item media doesn't arrive in time to continue playback.

`class let mediaSelectionDidChangeNotification: NSNotification.Name`

A notification the player item posts when its media selection changes.

`class let recommendedTimeOffsetFromLiveDidChangeNotification: NSNotification.Name`

A notification the player item posts when its offset from the live time changes.

`class let newAccessLogEntryNotification: NSNotification.Name`

A notification the system posts when a player item adds a new entry to its access log.

`class let newErrorLogEntryNotification: NSNotification.Name`

A notification the system posts when a player item adds a new entry to its error log.

## Managing time offsets

`var automaticallyPreservesTimeOffsetFromLive: Bool`

A Boolean value that indicates whether the player preserves its time offset from the live time after a buffering operation.

`var recommendedTimeOffsetFromLive: CMTime`

A recommended time offset from the live time based on observed network conditions.

`var configuredTimeOffsetFromLive: CMTime`

A time value that indicates the offset from the live time to start playback, or resume playback after a seek to positive infinity.

## Configuring presentation

`var presentationSize: CGSize`

The size at which the visual portion of the item is presented by the player.

`var preferredMaximumResolution: CGSize`

The desired maximum resolution of a video that is to be downloaded.

`var videoApertureMode: AVVideoApertureMode`

The video aperture mode to apply during playback.

`struct AVVideoApertureMode`

A value that describes how a video is scaled or cropped.

## Accessing Now Playing information

`var nowPlayingInfo: [String : Any]?`

The current now playing information for the player item.

## Configuring HDR settings

`var appliesPerFrameHDRDisplayMetadata: Bool`

A Boolean value that indicates whether the player item applies per-frame HDR display metadata during playback.

## Configuring video compositing

`var videoComposition: AVVideoComposition?`

The video composition settings to be applied during playback.

`var customVideoCompositor: (any AVVideoCompositing)?`

The custom video compositor.

`var seekingWaitsForVideoCompositionRendering: Bool`

A Boolean value that indicates whether the item's timing follows the displayed video frame when seeking with a video composition.

## Configuring audio

`var audioMix: AVAudioMix?`

The audio mix parameters to be applied during playback.

`var audioTimePitchAlgorithm: AVAudioTimePitchAlgorithm`

The processing algorithm used to manage audio pitch for scaled audio edits.

`var allowedAudioSpatializationFormats: AVAudioSpatializationFormats`

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

`struct AVAudioSpatializationFormats`

A structure that defines the spatialization formats that a player item supports.

~~`var isAudioSpatializationAllowed: Bool`~~

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

Deprecated

## Managing player item outputs

`var outputs: [AVPlayerItemOutput]`

An array of outputs associated with the player item.

`func add(AVPlayerItemOutput)`

Adds the specified player item output object to the receiver.

`func remove(AVPlayerItemOutput)`

Removes the specified player item output object from the receiver.

## Managing player item data collectors

`var mediaDataCollectors: [AVPlayerItemMediaDataCollector]`

The collection of associated media data collectors.

`func add(AVPlayerItemMediaDataCollector)`

Adds the specified media data collector to the player item's collection of media collectors.

`func remove(AVPlayerItemMediaDataCollector)`

Removes the specified media data collector from the player item's collection of media collectors.

## Configuring network behavior

`var preferredPeakBitRate: Double`

The desired limit, in bits per second, of network bandwidth consumption for this item.

`var preferredForwardBufferDuration: TimeInterval`

The duration the player should buffer media from the network ahead of the playhead to guard against playback disruption.

`var canUseNetworkResourcesForLiveStreamingWhilePaused: Bool`

A Boolean value that indicates whether the player item can use network resources to keep the playback state up to date while paused.

## Configuring player items for AVKit

`var navigationMarkerGroups: [AVNavigationMarkersGroup]`

The time marker groups that provide ways to navigate the player item's content.

`var nextContentProposal: AVContentProposal?`

The item proposed to follow the current content.

## Requesting playback authorization in tvOS

`func requestPlaybackRestrictionsAuthorization((Bool, (any Error)?) -> Void)`

Determines whether this item is subject to parental restrictions, and, if so, prompts the user to enter the restrictions passcode.

`func cancelPlaybackRestrictionsAuthorizationRequest()`

Cancels a pending authorization request and dismisses the passcode entry, if displayed.

## Managing playback authorization in macOS

`var isContentAuthorizedForPlayback: Bool`

A Boolean value that indicates whether the content has been authorized by the user.

`var isAuthorizationRequiredForPlayback: Bool`

A Boolean value that indicates whether authorization is required to play the content.

`var isApplicationAuthorizedForPlayback: Bool`

A Boolean value that indicates whether the application can be used to play the content.

`func requestContentAuthorizationAsynchronously(withTimeoutInterval: TimeInterval, completionHandler: () -> Void)`

Presents the user the opportunity to authorize the content for playback.

`var contentAuthorizationRequestStatus: AVContentAuthorizationStatus`

The status of the most recent content authorization request.

`enum AVContentAuthorizationStatus`

A value representing the status of a content authorization request.

`func cancelContentAuthorizationRequest()`

Cancels the currently outstanding content authorization request.

## Accessing initialization parameters

`var asset: AVAsset`

The asset provided during initialization.

`var automaticallyLoadedAssetKeys: [String]`

The array of asset keys to be automatically loaded before the player item is ready to play.

## Copying an player item

`func copy() -> Any`

Creates a copy of the object.

`func copy(with: NSZone?) -> Any`

Creates a copy of the object with the specified zone.

## Deprecated

Deprecated symbols

Review unsupported symbols and their replacements.

---

# Relationships

## Inherits From

`NSObject`

## Conforms To

`AVMetricEventStreamPublisher`
`CVarArg`
`Copyable`
`CustomDebugStringConvertible`
`CustomStringConvertible`
`Equatable`
`Hashable`
`NSCopying`
`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 AVPlayer`

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

`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.