MPMoviePlayerController Class Reference

Developer

Contents

2012-09-19 | Copyright © 2012 Apple Inc. All Rights Reserved.

2012-09-19 | Copyright © 2012 Apple Inc. All Rights Reserved.

2012-09-19 | Copyright © 2012 Apple Inc. All Rights Reserved.

MPMoviePlayerController Class Reference

Inherits from	NSObject
Conforms to	MPMediaPlayback NSObject (NSObject)
Framework	/System/Library/Frameworks/MediaPlayer.framework
Availability	Available in iOS 2.0 and later.
Companion guide	Multimedia Programming Guide
Declared in	MPMoviePlayerController.h
Related sample code	MoviePlayer

Overview

A movie player (of type `MPMoviePlayerController`) manages the playback of a movie from a file or a network stream. Playback occurs in a view owned by the movie player and takes place either fullscreen or inline. You can incorporate a movie player's view into a view hierarchy owned by your app, or use an `MPMoviePlayerViewController` object to manage the presentation for you.

Movie players (iOS 4.3 and later) support wireless movie playback to AirPlay-enabled hardware such as Apple TV. The movie player presents a control that allows the user to choose AirPlay-enabled hardware for playback when such hardware is in range. Starting in iOS 5.0, AirPlay playback is enabled by default. To disable AirPlay in your app, set the `allowsAirPlay` (page 12) property to `NO`.

When you add a movie player's view to your app's view hierarchy, be sure to size the frame correctly, as shown here:

```
MPMoviePlayerController *player =
        [[MPMoviePlayerController alloc] initWithContentURL: myURL];
[player prepareToPlay];
[player.view setFrame: myView.bounds];  // player's frame must match parent's
```

2012-09-19 | Copyright © 2012 Apple Inc. All Rights Reserved.

```
[myView addSubview: player.view];
// ...
[player play];
```

Consider a movie player view to be an opaque structure. You can add your own custom subviews to layer content on top of the movie but you must never modify any of its existing subviews.

In addition to layering content on top of a movie, you can provide custom background content by adding subviews to the view in the `backgroundView` (page 13) property. Custom subviews are supported in both inline and fullscreen playback modes but you must adjust the positions of your views when entering or exiting fullscreen mode. Use the `MPMoviePlayerWillEnterFullscreenNotification` and `MPMoviePlayerWillExitFullscreenNotification` notifications to detect changes to and from fullscreen mode.

This class supports programmatic control of movie playback, and user-based control via buttons supplied by the movie player. You can control most aspects of playback programmatically using the methods and properties of the `MPMediaPlayback` protocol, to which this class conforms. The methods and properties of that protocol let you start and stop playback, seek forward and backward through the movie's content, and even change the playback rate. In addition, the `controlStyle` (page 13) property of this class lets you display a set of standard system controls that allow the user to manipulate playback. You can also set the `shouldAutoplay` (page 20) property for network-based content to start automatically.

You typically specify the movie you want to play when you create a new `MPMoviePlayerController` object. However, you can also change the currently playing movie by changing the value in the `contentURL` (page 13) property. Changing this property lets you reuse the same movie player controller object in multiple places. For performance reasons you may want to play movies as local files. Do this by first downloading them to a local directory.

> **Note:** Although you can create multiple `MPMoviePlayerController` objects and present their views in your interface, only one movie player at a time can play its movie.

To facilitate the creation of video bookmarks or chapter links for a long movie, the `MPMoviePlayerController` class defines methods for generating thumbnail images at specific times within a movie. You can request a single thumbnail image using the `thumbnailImageAtTime:timeOption:` (page 24) method or request multiple thumbnail images using the `requestThumbnailImagesAtTimes:timeOption:` (page 23) method.

To play a network stream whose URL requires access credentials, first create an appropriate `NSURLCredential` object. Do this by calling, for example, the `initWithUser:password:persistence:` method, as shown here:

2012-09-19 | Copyright © 2012 Apple Inc. All Rights Reserved.

```
NSURLCredential *credential = [[NSURLCredential alloc]
                        initWithUser: @"userName"
                            password: @"password"
                         persistence: NSURLCredentialPersistenceForSession];

self.credential = credential;
[credential release];
```

In addition, create an appropriate `NSURLProtectionSpace` object, as shown here. Make appropriate modifications for the realm you are accessing:

```
NSURLProtectionSpace *protectionSpace = [[NSURLProtectionSpace alloc]
                             initWithHost: "@streams.mydomain.com"
                                     port: 80
                                 protocol: @"http"
                                    realm: @"mydomain.com"
                       authenticationMethod: NSURLAuthenticationMethodDefault];

self.protectionSpace = protectionSpace;
[protectionSpace release];
```

Add the URL credential and the protection space to the Singleton `NSURLCredentialStorage` object. Do this by calling, for example, the `setCredential:forProtectionSpace:` method, as shown here:

```
[[NSURLCredentialStorage sharedCredentialStorage]
                      setDefaultCredential: credential
                        forProtectionSpace: protectionSpace];
```

With the credential and protection space information in place, you can then play the protected stream.

Movie Player Notifications

A movie player generates notifications to keep your app informed about the state of movie playback. In addition to being notified when playback finishes, your app can be notified in the following situations:

- When the movie player begins playing, is paused, or begins seeking forward or backward
- When AirPlay playback starts or ends

2012-09-19 | Copyright © 2012 Apple Inc. All Rights Reserved.

- When the scaling mode of the movie changes
- When the movie enters or exits fullscreen mode
- When the load state for network-based movies changes
- When meta-information about the movie itself becomes available

For more information, see the Notifications section in this document.

Supported Formats

This class plays any movie or audio file supported in iOS. This includes both streamed content and fixed-length files. For movie files, this typically means files with the extensions `.mov`, `.mp4`, `.mpv`, and `.3gp` and using one of the following compression standards:

- H.264 Baseline Profile Level 3.0 video, up to 640 x 480 at 30 fps. (The Baseline profile does not support B frames.)
- MPEG-4 Part 2 video (Simple Profile)

If you use this class to play audio files, it displays a white screen with a QuickTime logo while the audio plays. For audio files, this class supports AAC-LC audio at up to 48 kHz, and MP3 (MPEG-1 Audio Layer 3) up to 48 kHz, stereo audio.

Behavior in iOS 4.3 and Earlier

In iOS 4.3 and earlier, a new movie player was automatically prepared to play. Starting in iOS 5.0, in order to facilitate finer-grained playback control, a new movie player is not automatically prepared to play. See the discussion for the `initWithContentURL:` (page 22) instance method.

Also, AirPlay was disabled by default in iOS 4.3 and earlier. Starting in iOS 5.0, the default for the `allowsAirPlay` (page 12) property is `YES`.

Behavior in iOS 3.1 and Earlier

In iOS 3.1 and earlier, this class implemented a full-screen movie player only. After creating the movie player and initializing it with a single movie file, you called the `play` method to present the movie. (The definition of the `play` method has since moved out of this class and into the `MPMediaPlayback` protocol.) The movie player object itself handled the actual presentation of the movie content.

2012-09-19 | Copyright © 2012 Apple Inc. All Rights Reserved.

Tasks

Creating and Initializing the Object

– `initWithContentURL:` (page 22)
Returns a `MPMoviePlayerController` object initialized with the movie at the specified URL.

Accessing Movie Properties

`contentURL` (page 13) *property*
The URL that points to the movie file.

`movieSourceType` (page 17) *property*
The playback type of the movie.

`movieMediaTypes` (page 17) *property*
The types of media available in the movie. (read-only)

`allowsAirPlay` (page 12) *property*
Specifies whether the movie player allows AirPlay movie playback.

`airPlayVideoActive` (page 12) *property*
Indicates whether the movie player is currently playing video via AirPlay.

`naturalSize` (page 17) *property*
The width and height of the movie frame. (read-only)

`fullscreen` (page 15) *property*
A Boolean that indicates whether the movie player is in full-screen mode.

– `setFullscreen:animated:` (page 23)
Causes the movie player to enter or exit full-screen mode.

`scalingMode` (page 20) *property*
The scaling mode to use when displaying the movie.

`controlStyle` (page 13) *property*
The style of the playback controls.

`useApplicationAudioSession` (page 42) *property* Deprecated in iOS 6.0
A Boolean value that indicates whether the movie player should use the app's audio session. (Deprecated. There is not replacement for this property and its use is discouraged.)

2012-09-19 | Copyright © 2012 Apple Inc. All Rights Reserved.

Accessing the Movie Duration

`duration` (page 14) *property*

The duration of the movie, measured in seconds. (read-only)

`playableDuration` (page 18) *property*

The amount of currently playable content. (read-only)

Accessing the View

`view` (page 21) *property*

The view containing the movie content and controls. (read-only)

`backgroundView` (page 13) *property*

A customizable view that is displayed behind the movie content. (read-only)

Controlling and Monitoring Playback

See also the methods of the `MPMediaPlayback` protocol.

`loadState` (page 16) *property*

The network load state of the movie player. (read-only)

`playbackState` (page 18) *property*

The current playback state of the movie player. (read-only)

`initialPlaybackTime` (page 16) *property*

The time, specified in seconds within the video timeline, when playback should start.

`endPlaybackTime` (page 14) *property*

The end time (measured in seconds) for playback of the movie.

`shouldAutoplay` (page 20) *property*

A Boolean that indicates whether a movie should begin playback automatically.

`readyForDisplay` (page 19) *property*

A Boolean that indicates whether the first video frame of the movie is ready to be displayed.

`repeatMode` (page 19) *property*

Determines how the movie player repeats the playback of the movie.

– `timedMetadata` (page 24)

Obtains the most recent time-based metadata provided by the streamed movie.

2012-09-19 | Copyright © 2012 Apple Inc. All Rights Reserved.

Generating Thumbnail Images

– `thumbnailImageAtTime:timeOption:` (page 24)
Captures and returns a thumbnail image from the current movie.

– `requestThumbnailImagesAtTimes:timeOption:` (page 23)
Captures one or more thumbnail images asynchronously from the current movie.

– `cancelAllThumbnailImageRequests` (page 21)
Cancels all pending asynchronous thumbnail image requests.

Retrieving Movie Logs

`accessLog` (page 11) *property*
A snapshot of the network playback log for the movie player if it is playing a network stream.

`errorLog` (page 15) *property*
A snapshot of the playback failure error log for the movie player if it is playing a network stream.

Unavailable Methods and Properties

The following methods and properties are no longer available as of iOS 3.2 and must not be used.

`backgroundColor` (page 41) *property* Available in iOS 2.0 through iOS 3.1
The color of the background area behind the movie. (Deprecated. Get the view from the `backgroundView` (page 13) property and set its color directly.)

`movieControlMode` (page 41) *property* Available in iOS 2.0 through iOS 3.1
The user controls to display. (Deprecated. Use the `controlStyle` (page 13) property instead.)

Properties

accessLog

A snapshot of the network playback log for the movie player if it is playing a network stream.

```
@property (nonatomic, readonly) MPMovieAccessLog *accessLog
```

Discussion
Can be `nil`. For information about movie access logs, refer to *MPMovieAccessLog Class Reference*.

2012-09-19 | Copyright © 2012 Apple Inc. All Rights Reserved.

Availability
Available in iOS 4.3 and later.

Declared in
`MPMoviePlayerController.h`

airPlayVideoActive

Indicates whether the movie player is currently playing video via AirPlay.

```
@property (nonatomic, readonly, getter=isAirPlayVideoActive) BOOL airPlayVideoActive
```

Discussion
You can query this property after receiving an `MPMoviePlayerIsAirPlayVideoActiveDidChangeNotification` (page 36) notification to find out whether the AirPlay video started or stopped.

Availability
Available in iOS 5.0 and later.

Declared in
`MPMoviePlayerController.h`

allowsAirPlay

Specifies whether the movie player allows AirPlay movie playback.

```
@property (nonatomic) BOOL allowsAirPlay
```

Discussion
A movie player supports wireless movie playback to AirPlay-enabled hardware. By default, this property's value is `YES`.

To disable AirPlay movie playback, set this property's value to `NO`. The movie player then presents a control that allows the user to choose AirPlay-enabled hardware for playback when such hardware is in range.

Availability
Available in iOS 4.3 and later.

Related Sample Code
MoviePlayer

Declared in
`MPMoviePlayerController.h`

backgroundView

A customizable view that is displayed behind the movie content. (read-only)

```
@property (nonatomic, readonly) UIView *backgroundView
```

Discussion
This view provides the backing content, on top of which the movie content is displayed. You can add subviews to the background view if you want to display custom background content.

This view is part of the view hierarchy returned by the `view` (page 21) property.

Availability
Available in iOS 3.2 and later.

Related Sample Code
MoviePlayer

Declared in
`MPMoviePlayerController.h`

contentURL

The URL that points to the movie file.

```
@property (nonatomic, copy) NSURL *contentURL
```

Discussion
If you set this property while a movie is playing, that movie pauses and the new movie begins loading. The new movie starts playing at the beginning.

Availability
Available in iOS 2.0 and later.

Declared in
`MPMoviePlayerController.h`

controlStyle

The style of the playback controls.

2012-09-19 | Copyright © 2012 Apple Inc. All Rights Reserved.

```
@property (nonatomic) MPMovieControlStyle controlStyle
```

Discussion
The default value of this property is `MPMovieControlStyleDefault` (page 26). You can change the value of this property to change the style of the controls or to hide the controls altogether. For a list of available control styles, see "MPMovieControlStyle" (page 26).

Availability
Available in iOS 3.2 and later.

Related Sample Code
MoviePlayer

Declared in
`MPMoviePlayerController.h`

duration

The duration of the movie, measured in seconds. (read-only)

```
@property (nonatomic, readonly) NSTimeInterval duration
```

Discussion
If the duration of the movie is not known, the value in this property is `0.0`. If the duration is subsequently determined, this property is updated and a `MPMovieDurationAvailableNotification` (page 34) notification is posted.

Availability
Available in iOS 3.2 and later.

Declared in
`MPMoviePlayerController.h`

endPlaybackTime

The end time (measured in seconds) for playback of the movie.

```
@property (nonatomic) NSTimeInterval endPlaybackTime
```

Discussion
The default value of this property is -1, which indicates the natural end time of the movie. This property is not applicable for streamed content.

2012-09-19 | Copyright © 2012 Apple Inc. All Rights Reserved.

Availability
Available in iOS 3.2 and later.

Declared in
`MPMoviePlayerController.h`

errorLog

A snapshot of the playback failure error log for the movie player if it is playing a network stream.

```
@property (nonatomic, readonly) MPMovieErrorLog *errorLog
```

Discussion
Can be `nil`. For information about movie error logs, refer to *MPMovieErrorLog Class Reference*.

Availability
Available in iOS 4.3 and later.

Declared in
`MPMoviePlayerController.h`

fullscreen

A Boolean that indicates whether the movie player is in full-screen mode.

```
@property (nonatomic, getter=isFullscreen) BOOL fullscreen
```

Discussion
The default value of this property is `NO`. Changing the value of this property causes the movie player to enter or exit full-screen mode immediately. If you want to animate the transition to full-screen mode, use the `setFullscreen:animated:` method instead.

Whenever the movie player enters or exits full-screen mode, it posts appropriate notifications to reflect the change. For example, upon entering full-screen mode, it posts `MPMoviePlayerWillEnterFullscreenNotification` (page 39) and `MPMoviePlayerDidEnterFullscreenNotification` (page 35) notifications. Upon exiting from full-screen mode, it posts `MPMoviePlayerWillExitFullscreenNotification` (page 39) and `MPMoviePlayerDidExitFullscreenNotification` (page 36) notifications.

The value of this property may also change as a result of the user interacting with the movie player controls.

Availability
Available in iOS 3.2 and later.

2012-09-19 | Copyright © 2012 Apple Inc. All Rights Reserved.

See Also
– `setFullscreen:animated:` (page 23)

Declared in
`MPMoviePlayerController.h`

initialPlaybackTime

The time, specified in seconds within the video timeline, when playback should start.

```
@property (nonatomic) NSTimeInterval initialPlaybackTime
```

Discussion
For progressively downloaded content, playback starts at the closest key frame prior to the provided time. For video-on-demand content, playback starts at the nearest segment boundary to the provided time. For live video streams, the playback start time is measured from the start of the current playlist and is rounded to the nearest segment boundary.

The default value of this property is -1, which indicates the natural start time of the movie.

Availability
Available in iOS 3.0 and later.

Declared in
`MPMoviePlayerController.h`

loadState

The network load state of the movie player. (read-only)

```
@property (nonatomic, readonly) MPMovieLoadState loadState
```

Discussion
See the "MPMovieLoadState" (page 25) enumeration for possible values of this property. To be notified of changes to the load state of a movie player, register for the `MPMoviePlayerLoadStateDidChangeNotification` (page 36) notification.

Availability
Available in iOS 3.2 and later.

Related Sample Code
MoviePlayer

Declared in
`MPMoviePlayerController.h`

movieMediaTypes

The types of media available in the movie. (read-only)

```
@property (nonatomic, readonly) MPMovieMediaTypeMask movieMediaTypes
```

Discussion
Movies can contain a combination of audio, video, or a combination of the two. The default value of this property is `MPMovieMediaTypeMaskNone` (page 30). See the "MPMovieMediaTypeMask" (page 30) enumeration for possible values of this property.

Availability
Available in iOS 3.2 and later.

Declared in
`MPMoviePlayerController.h`

movieSourceType

The playback type of the movie.

```
@property (nonatomic) MPMovieSourceType movieSourceType
```

Discussion
The default value of this property is `MPMovieSourceTypeUnknown` (page 31). This property provides a clue to the playback system as to how it should download and buffer the movie content. If you know the source type of the movie, setting the value of this property before playback begins can improve the load times for the movie content. If you do not set the source type explicitly before playback, the movie player controller must gather this information, which might delay playback.

Availability
Available in iOS 3.2 and later.

Declared in
`MPMoviePlayerController.h`

naturalSize

The width and height of the movie frame. (read-only)

2012-09-19 | Copyright © 2012 Apple Inc. All Rights Reserved.

```
@property (nonatomic, readonly) CGSize naturalSize
```

Discussion
This property reports the clean aperture of the video in square pixels. Thus, the reported dimensions take into account anamorphic content and aperture modes.

It is possible for the natural size of a movie to change during playback. This typically happens when the bit-rate of streaming content changes or when playback toggles between audio-only and a combination of audio and video.

Availability
Available in iOS 3.2 and later.

Declared in
`MPMoviePlayerController.h`

playableDuration

The amount of currently playable content. (read-only)

```
@property (nonatomic, readonly) NSTimeInterval playableDuration
```

Discussion
For progressively downloaded network content, this property reflects the amount of content that can be played now.

Availability
Available in iOS 3.2 and later.

Declared in
`MPMoviePlayerController.h`

playbackState

The current playback state of the movie player. (read-only)

```
@property (nonatomic, readonly) MPMoviePlaybackState playbackState
```

Discussion
The playback state is affected by programmatic calls to play, pause, or stop the movie player. It can also be affected by user interactions or by the network, in cases where streaming content cannot be buffered fast enough.

2012-09-19 | Copyright © 2012 Apple Inc. All Rights Reserved.

See the "MPMoviePlaybackState" (page 27) enumeration for possible values of this property. To be notified of changes to the playback state of a movie player, register for the MPMoviePlayerPlaybackStateDidChangeNotification (page 37) notification.

Availability
Available in iOS 3.2 and later.

Related Sample Code
MoviePlayer

Declared in
MPMoviePlayerController.h

readyForDisplay

A Boolean that indicates whether the first video frame of the movie is ready to be displayed.

```
@property(nonatomic, readonly) BOOL readyForDisplay
```

Discussion
The default value of this property is NO. This property returns YES if the first video frame is ready to be displayed and returns NO if there are no video tracks associated. When the value of this property changes to YES, a MPMoviePlayerReadyForDisplayDidChangeNotification (page 40) is sent.

Availability
Available in iOS 6.0 and later.

Declared in
MPMoviePlayerController.h

repeatMode

Determines how the movie player repeats the playback of the movie.

```
@property (nonatomic) MPMovieRepeatMode repeatMode
```

Discussion
The default value of this property is MPMovieRepeatModeNone (page 28). For a list of available repeat modes, see "MPMovieRepeatMode" (page 28).

Availability
Available in iOS 3.2 and later.

2012-09-19 | Copyright © 2012 Apple Inc. All Rights Reserved.

Related Sample Code
MoviePlayer

Declared in
`MPMoviePlayerController.h`

scalingMode

The scaling mode to use when displaying the movie.

```
@property (nonatomic) MPMovieScalingMode scalingMode
```

Discussion
Changing this property while the movie player is visible causes the current movie to animate to the new scaling mode.

The default value of this property is `MPMovieScalingModeAspectFit`. For a list of available scaling modes, see "MPMovieScalingMode" (page 29).

Availability
Available in iOS 2.0 and later.

Related Sample Code
MoviePlayer

Declared in
`MPMoviePlayerController.h`

shouldAutoplay

A Boolean that indicates whether a movie should begin playback automatically.

```
@property (nonatomic) BOOL shouldAutoplay
```

Discussion
The default value of this property is `YES`. This property determines whether the playback of network-based content begins automatically when there is enough buffered data to ensure uninterrupted playback.

Availability
Available in iOS 3.2 and later.

Declared in
`MPMoviePlayerController.h`

2012-09-19 | Copyright © 2012 Apple Inc. All Rights Reserved.

view

The view containing the movie content and controls. (read-only)

```
@property (nonatomic, readonly) UIView *view
```

Discussion
This property contains the view used for presenting the video content. This view incorporates all the background, content, and controls needed to display movies. You can incorporate this view into your own view hierarchies or present it by itself using a view controller.

To embed the view into your own view hierarchies, add it as a subview to one of your existing views. A good place to do this is in the `loadView` or `viewDidLoad` method of the custom view controller that presents your view hierarchy. You are free to change the view's `frame` rectangle to accommodate the space available in your view hierarchy. The movie player uses the value in the `scalingMode` (page 20) property to scale the movie content to match the frame you specify.

If you want to present the view by itself—that is, without embedding it in an existing view hierarchy—you can use an instance of the `MPMoviePlayerViewController` class to manage the presentation of the view. That class works directly with the movie player controller to present the view by itself.

You can add subviews to the view in this property. You might do this in cases where you want to display custom playback controls or add other custom content that is relevant to your app.

Availability
Available in iOS 3.2 and later.

See Also
`@property backgroundView` (page 13)

Related Sample Code
MoviePlayer

Declared in
`MPMoviePlayerController.h`

Instance Methods

cancelAllThumbnailImageRequests

Cancels all pending asynchronous thumbnail image requests.

```
- (void)cancelAllThumbnailImageRequests
```

2012-09-19 | Copyright © 2012 Apple Inc. All Rights Reserved.

Discussion

This method cancels only requests made using the `requestThumbnailImagesAtTimes:timeOption:` (page 23) method. It does not cancel requests made synchronously using the `thumbnailImageAtTime:timeOption:` (page 24) method.

Availability

Available in iOS 3.2 and later.

Declared in

`MPMoviePlayerController.h`

initWithContentURL:

Returns a `MPMoviePlayerController` object initialized with the movie at the specified URL.

```
- (id)initWithContentURL:(NSURL *)url
```

Parameters

`url`

The location of the movie file. This file must be located either in your app directory or on a remote server.

Return Value

The movie player object.

Discussion

This method initializes a movie player, but does not prepare it for playback. To prepare a new movie player for playback, call the `prepareToPlay` method, described in *MPMediaPlayback Protocol Reference*.

To be notified when a new movie player is ready to play, register for the `MPMoviePlayerLoadStateDidChangeNotification` (page 36) notification. You can then check load state by accessing the `loadState` (page 16) property.

To check for errors in URL loading, register for the `MPMoviePlayerPlaybackDidFinishNotification` (page 37) notification. On error, this notification contains an `NSError` object available using the `@"error"` key in the notification's `userInfo` dictionary.

Availability

Available in iOS 2.0 and later.

Declared in

`MPMoviePlayerController.h`

2012-09-19 | Copyright © 2012 Apple Inc. All Rights Reserved.

requestThumbnailImagesAtTimes:timeOption:

Captures one or more thumbnail images asynchronously from the current movie.

```
- (void)requestThumbnailImagesAtTimes:(NSArray *)playbackTimes
timeOption:(MPMovieTimeOption)option
```

Parameters

`playbackTimes`

An array of `NSNumber` objects containing the times at which to capture the thumbnail images. Each time value represents the number of seconds from the beginning of the current movie.

`option`

The option to use when determining which specific frame to use for each thumbnail image. For a list of possible values, see "MPMovieTimeOption" (page 30).

Discussion

This method processes each thumbnail request separately and asynchronously. When the results for a single image arrive, the movie player posts a `MPMoviePlayerThumbnailImageRequestDidFinishNotification` (page 38) notification with the results for that image. Notifications are posted regardless of whether the image capture was successful or failed. You should register for this notification prior to calling this method.

Availability

Available in iOS 3.2 and later.

Declared in

`MPMoviePlayerController.h`

setFullscreen:animated:

Causes the movie player to enter or exit full-screen mode.

```
- (void)setFullscreen:(BOOL)fullscreen
animated:(BOOL)animated
```

Parameters

`fullscreen`

Specify `YES` to enter full-screen mode or `NO` to exit full-screen mode.

`animated`

Specify `YES` to animate the transition between modes or `NO` to switch immediately to the new mode.

Availability

Available in iOS 3.2 and later.

2012-09-19 | Copyright © 2012 Apple Inc. All Rights Reserved.

See Also

`@property fullscreen` (page 15)

Declared in

`MPMoviePlayerController.h`

thumbnailImageAtTime:timeOption:

Captures and returns a thumbnail image from the current movie.

```
- (UIImage *)thumbnailImageAtTime:(NSTimeInterval)playbackTime
timeOption:(MPMovieTimeOption)option
```

Parameters

`playbackTime`

The time at which to capture the thumbnail image. The time value represents the number of seconds from the beginning of the current movie.

`option`

The option to use when determining which specific frame to use for the thumbnail image. For a list of possible values, see "MPMovieTimeOption" (page 30).

Return Value

An image object containing the image from the movie or `nil` if the thumbnail could not be captured.

Discussion

This method captures the thumbnail image synchronously from the current movie (which is accessible from the `MPMovieSourceTypeUnknown` (page 31) property).

Availability

Available in iOS 3.2 and later.

Declared in

`MPMoviePlayerController.h`

timedMetadata

Obtains the most recent time-based metadata provided by the streamed movie.

```
- (NSArray *)timedMetadata
```

Return Value

An array of the most recent `MPTimedMetadata` objects provided by the streamed movie.

2012-09-19 | Copyright © 2012 Apple Inc. All Rights Reserved.

Availability
Available in iOS 4.0 and later.

Declared in
`MPMoviePlayerController.h`

Constants

MPMovieLoadState

Constants describing the network load state of the movie player.

```
enum {
   MPMovieLoadStateUnknown        = 0,
   MPMovieLoadStatePlayable       = 1 << 0,
   MPMovieLoadStatePlaythroughOK  = 1 << 1,
   MPMovieLoadStateStalled        = 1 << 2,
};
typedef NSInteger MPMovieLoadState;
```

Constants

`MPMovieLoadStateUnknown`

The load state is not known.

Available in iOS 3.2 and later.

Declared in `MPMoviePlayerController.h`.

`MPMovieLoadStatePlayable`

The buffer has enough data that playback can begin, but it may run out of data before playback finishes.

Available in iOS 3.2 and later.

Declared in `MPMoviePlayerController.h`.

`MPMovieLoadStatePlaythroughOK`

Enough data has been buffered for playback to continue uninterrupted.

Available in iOS 3.2 and later.

Declared in `MPMoviePlayerController.h`.

`MPMovieLoadStateStalled`

The buffering of data has stalled. If started now, playback may pause automatically if the player runs out of buffered data.

Available in iOS 3.2 and later.

Declared in `MPMoviePlayerController.h`.

2012-09-19 | Copyright © 2012 Apple Inc. All Rights Reserved.

MPMovieControlStyle

Constants describing the style of the playback controls.

```
enum {
   MPMovieControlStyleNone,
   MPMovieControlStyleEmbedded,
   MPMovieControlStyleFullscreen,
   MPMovieControlStyleDefault = MPMovieControlStyleFullscreen
};
typedef NSInteger MPMovieControlStyle;
```

Constants

`MPMovieControlStyleNone`

No controls are displayed.

Available in iOS 3.2 and later.

Declared in `MPMoviePlayerController.h`.

`MPMovieControlStyleEmbedded`

Controls for an embedded view are displayed. The controls include a start/pause button, a scrubber bar, and a button for toggling between fullscreen and embedded display modes.

Available in iOS 3.2 and later.

Declared in `MPMoviePlayerController.h`.

`MPMovieControlStyleFullscreen`

Controls for fullscreen playback are displayed. The controls include a start/pause button, a scrubber bar, forward and reverse seeking buttons, a button for toggling between fullscreen and embedded display modes, a button for toggling the aspect fill mode, and a Done button. Tapping the done button pauses the video and exits fullscreen mode.

Available in iOS 3.2 and later.

Declared in `MPMoviePlayerController.h`.

`MPMovieControlStyleDefault`

Fullscreen controls are displayed by default.

Available in iOS 3.2 and later.

Declared in `MPMoviePlayerController.h`.

MPMovieFinishReason

Constants describing the reason that playback ended.

```
enum {
```

2012-09-19 | Copyright © 2012 Apple Inc. All Rights Reserved.

```
   MPMovieFinishReasonPlaybackEnded,
   MPMovieFinishReasonPlaybackError,
   MPMovieFinishReasonUserExited
};
typedef NSInteger MPMovieFinishReason;
```

Constants

`MPMovieFinishReasonPlaybackEnded`

The end of the movie was reached.

Available in iOS 3.2 and later.

Declared in `MPMoviePlayerController.h`.

`MPMovieFinishReasonPlaybackError`

There was an error during playback.

Available in iOS 3.2 and later.

Declared in `MPMoviePlayerController.h`.

`MPMovieFinishReasonUserExited`

The user stopped playback.

Available in iOS 3.2 and later.

Declared in `MPMoviePlayerController.h`.

MPMoviePlaybackState

Constants describing the current playback state of the movie player.

```
enum {
   MPMoviePlaybackStateStopped,
   MPMoviePlaybackStatePlaying,
   MPMoviePlaybackStatePaused,
   MPMoviePlaybackStateInterrupted,
   MPMoviePlaybackStateSeekingForward,
   MPMoviePlaybackStateSeekingBackward
};
typedef NSInteger MPMoviePlaybackState;
```

Constants

`MPMoviePlaybackStateStopped`

Playback is currently stopped. Playback will commence from the beginning of the movie.

Available in iOS 3.2 and later.

Declared in `MPMoviePlayerController.h`.

2012-09-19 | Copyright © 2012 Apple Inc. All Rights Reserved.

`MPMoviePlaybackStatePlaying`

Playback is currently under way.

Available in iOS 3.2 and later.

Declared in `MPMoviePlayerController.h`.

`MPMoviePlaybackStatePaused`

Playback is currently paused. Playback will resume from the point where it was paused.

Available in iOS 3.2 and later.

Declared in `MPMoviePlayerController.h`.

`MPMoviePlaybackStateInterrupted`

Playback is temporarily interrupted, perhaps because the buffer ran out of content.

Available in iOS 3.2 and later.

Declared in `MPMoviePlayerController.h`.

`MPMoviePlaybackStateSeekingForward`

The movie player is currently seeking towards the end of the movie.

Available in iOS 3.2 and later.

Declared in `MPMoviePlayerController.h`.

`MPMoviePlaybackStateSeekingBackward`

The movie player is currently seeking towards the beginning of the movie.

Available in iOS 3.2 and later.

Declared in `MPMoviePlayerController.h`.

MPMovieRepeatMode

Constants describing how the movie player repeats content at the end of playback.

```
enum {
   MPMovieRepeatModeNone,
   MPMovieRepeatModeOne
};
typedef NSInteger MPMovieRepeatMode;
```

Constants

`MPMovieRepeatModeNone`

Content is not repeated when playback finishes

Available in iOS 3.2 and later.

Declared in `MPMoviePlayerController.h`.

2012-09-19 | Copyright © 2012 Apple Inc. All Rights Reserved.

`MPMovieRepeatModeOne`

The current movie is repeated when it finishes.

Available in iOS 3.2 and later.

Declared in `MPMoviePlayerController.h`.

MPMovieScalingMode

Constants describing how the movie content is scaled to fit the frame of its view.

```
typedef enum {
   MPMovieScalingModeNone,
   MPMovieScalingModeAspectFit,
   MPMovieScalingModeAspectFill,
   MPMovieScalingModeFill
} MPMovieScalingMode;
```

Constants

`MPMovieScalingModeNone`

Do not scale the movie.

Available in iOS 2.0 and later.

Declared in `MPMoviePlayerController.h`.

`MPMovieScalingModeAspectFit`

Scale the movie uniformly until one dimension fits the visible bounds of the view exactly. In the other dimension, the region between the edge of the movie and the edge of the view is filled with a black bar. The aspect ratio of the movie is preserved.

Available in iOS 2.0 and later.

Declared in `MPMoviePlayerController.h`.

`MPMovieScalingModeAspectFill`

Scale the movie uniformly until the movie fills the visible bounds of the view. Content at the edges of the larger of the two dimensions is clipped so that the other dimension fits the view exactly. The aspect ratio of the movie is preserved.

Available in iOS 2.0 and later.

Declared in `MPMoviePlayerController.h`.

`MPMovieScalingModeFill`

Scale the movie until both dimensions fit the visible bounds of the view exactly. The aspect ratio of the movie is not preserved.

Available in iOS 2.0 and later.

Declared in `MPMoviePlayerController.h`.

2012-09-19 | Copyright © 2012 Apple Inc. All Rights Reserved.

MPMovieTimeOption

Constants describing which frame to use when generating thumbnail images.

```
enum {
   MPMovieTimeOptionNearestKeyFrame,
   MPMovieTimeOptionExact
};
typedef NSInteger MPMovieTimeOption;
```

Constants

`MPMovieTimeOptionNearestKeyFrame`

Generate a thumbnail image using the nearest key frame. This frame could be several frames away from the current frame. This option generally offers better performance than trying to find the exact frame.

Available in iOS 3.2 and later.

Declared in `MPMoviePlayerController.h`.

`MPMovieTimeOptionExact`

Use the exact current frame.

Available in iOS 3.2 and later.

Declared in `MPMoviePlayerController.h`.

MPMovieMediaTypeMask

Specifies the types of content available in the movie file.

```
enum {
   MPMovieMediaTypeMaskNone  = 0,
   MPMovieMediaTypeMaskVideo = 1 << 0,
   MPMovieMediaTypeMaskAudio = 1 << 1
};
typedef NSInteger MPMovieMediaTypeMask;
```

Constants

`MPMovieMediaTypeMaskNone`

The types of media available in the media are not yet known.

Available in iOS 3.2 and later.

Declared in `MPMoviePlayerController.h`.

2012-09-19 | Copyright © 2012 Apple Inc. All Rights Reserved.

`MPMovieMediaTypeMaskVideo`

The movie file contains video media.

Available in iOS 3.2 and later.

Declared in `MPMoviePlayerController.h`.

`MPMovieMediaTypeMaskAudio`

The movie file contains audio media.

Available in iOS 3.2 and later.

Declared in `MPMoviePlayerController.h`.

Discussion

You can OR the specified constants together to specify a movie

MPMovieSourceType

Specifies the type of the movie file.

```
enum {
   MPMovieSourceTypeUnknown,
   MPMovieSourceTypeFile,
   MPMovieSourceTypeStreaming
};
typedef NSInteger MPMovieSourceType;
```

Constants

`MPMovieSourceTypeUnknown`

The movie type is not yet known.

Available in iOS 3.2 and later.

Declared in `MPMoviePlayerController.h`.

`MPMovieSourceTypeFile`

The movie is a local file or is a file that can be downloaded from the network.

Available in iOS 3.2 and later.

Declared in `MPMoviePlayerController.h`.

`MPMovieSourceTypeStreaming`

The movie is a live or on-demand stream.

Available in iOS 3.2 and later.

Declared in `MPMoviePlayerController.h`.

2012-09-19 | Copyright © 2012 Apple Inc. All Rights Reserved.

Thumbnail Notification User Info Keys

The following keys may be found in the `userInfo` dictionary of a `MPMoviePlayerThumbnailImageRequestDidFinishNotification` (page 38) notification.

```
NSString *const MPMoviePlayerThumbnailImageKey;
NSString *const MPMoviePlayerThumbnailTimeKey;
NSString *const MPMoviePlayerThumbnailErrorKey;
```

Constants

`MPMoviePlayerThumbnailImageKey`

The value of this key is a `UIImage` object containing the image that was obtained for the desired frame.

Available in iOS 3.2 and later.

Declared in `MPMoviePlayerController.h`.

`MPMoviePlayerThumbnailTimeKey`

The value of this key is a `NSNumber` object containing a double value. This value represents the actual time (measured in seconds) from the beginning of the movie at which the image was captured.

Available in iOS 3.2 and later.

Declared in `MPMoviePlayerController.h`.

`MPMoviePlayerThumbnailErrorKey`

The value of this key is an `NSError` object identifying the error that occurred, if any.

Available in iOS 3.2 and later.

Declared in `MPMoviePlayerController.h`.

Fullscreen Notification Keys

The following keys may be found in the `userInfo` dictionary of notifications for transitioning in or out of full-screen mode.

```
NSString *const MPMoviePlayerFullscreenAnimationDurationUserInfoKey;
NSString *const MPMoviePlayerFullscreenAnimationCurveUserInfoKey;
```

Constants

`MPMoviePlayerFullscreenAnimationDurationUserInfoKey`

The value of this key is an `NSNumber` containing a double value. This value represents the duration (measured in seconds) of the animation used to transition in or out of full-screen mode.

Available in iOS 3.2 and later.

Declared in `MPMoviePlayerController.h`.

2012-09-19 | Copyright © 2012 Apple Inc. All Rights Reserved.

`MPMoviePlayerFullscreenAnimationCurveUserInfoKey`

The value of this key is an `NSNumber` containing an integer value that represents one of the `UIViewAnimationCurve` constants.

Available in iOS 3.2 and later.

Declared in `MPMoviePlayerController.h`.

Playback Finished Notification Key

The following key may be found in the userInfo dictionary of a `MPMoviePlayerPlaybackDidFinishNotification` (page 37) notification.

```
NSString *const MPMoviePlayerPlaybackDidFinishReasonUserInfoKey;
```

Constants

`MPMoviePlayerPlaybackDidFinishReasonUserInfoKey`

The value of this key is an `NSNumber` containing an integer value that represents one of the `"MPMovieFinishReason"` (page 26) constants.

Available in iOS 3.2 and later.

Declared in `MPMoviePlayerController.h`.

MPMovieControlMode

Options for displaying movie playback controls. (Deprecated. Use the `"MPMovieControlStyle"` (page 26) constants in conjunction with the `controlStyle` (page 13) property instead.)

```
typedef enum {
   MPMovieControlModeDefault,
   MPMovieControlModeVolumeOnly,
   MPMovieControlModeHidden
} MPMovieControlMode;
```

Constants

`MPMovieControlModeDefault`

Display the standard controls for controlling playback. This includes play/pause controls, a volume slider, and a timeline control.

Available in iOS 2.0 and later.

Declared in `MPMoviePlayerController.h`.

2012-09-19 | Copyright © 2012 Apple Inc. All Rights Reserved.

`MPMovieControlModeVolumeOnly`

Display volume controls only.

Available in iOS 2.0 and later.

Declared in `MPMoviePlayerController.h`.

`MPMovieControlModeHidden`

Do not display any controls. This mode prevents the user from controlling playback.

Available in iOS 2.0 and later.

Declared in `MPMoviePlayerController.h`.

Notifications

To be notified of changes in movie player state such as playability, or changes in the availability of movie information such as duration, register for the appropriate notification. For all movie player notifications, the movie player whose state has changed is available as the object associated with the notification.

MPMovieDurationAvailableNotification

Posted when the duration of a movie has been determined. There is no `userInfo` dictionary.

To retrieve the duration of a movie, access the movie player's `duration` (page 14) property. The movie player whose state has changed is available as the object associated with the notification.

Availability
Available in iOS 3.2 and later.

Declared in
`MPMoviePlayerController.h`

MPMovieMediaTypesAvailableNotification

Posted when the available media types in a movie are determined. There is no `userInfo` dictionary.

To retrieve the available media types in a movie—audio, video, or both—access the movie player's `movieMediaTypes` (page 17) property. The movie player whose state has changed is available as the object associated with the notification.

Availability
Available in iOS 3.2 and later.

2012-09-19 | Copyright © 2012 Apple Inc. All Rights Reserved.

Declared in
`MPMoviePlayerController.h`

MPMovieNaturalSizeAvailableNotification

Posted when the natural frame size of a movie is first determined or subsequently changes. There is no `userInfo` dictionary.

To retrieve the natural frame size for a movie, access the movie player's `naturalSize` (page 17) property. The movie player whose state has changed is available as the object associated with the notification.

Availability
Available in iOS 3.2 and later.

Declared in
`MPMoviePlayerController.h`

MPMoviePlayerContentPreloadDidFinishNotification

Posted when a movie is in memory and ready to play. The movie player whose state has changed is available as the object associated with the notification. If an error occurred during loading, the `userInfo` dictionary of this notification contains a key with the name "error" whose value is the `NSError` object describing the problem. (#Deprecated. Use the `MPMoviePlayerLoadStateDidChangeNotification` (page 36) notification to determine the readiness of the player.)

Availability
Available in iOS 2.0 and later.

Deprecated in iOS 3.2.

Declared in
`MPMoviePlayerController.h`

MPMoviePlayerDidEnterFullscreenNotification

Posted when a movie player has entered full-screen mode. There is no `userInfo` dictionary.

A movie player can enter full screen mode programmatically (see the `setFullscreen:animated:` (page 23) method) or by user interaction. The movie player whose state has changed is available as the object associated with the notification.

2012-09-19 | Copyright © 2012 Apple Inc. All Rights Reserved.

Availability
Available in iOS 3.2 and later.

Declared in
`MPMoviePlayerController.h`

MPMoviePlayerDidExitFullscreenNotification

Posted when a movie player has exited full-screen mode. There is no `userInfo` dictionary.

A movie player can exit full screen mode programmatically (see the `setFullscreen:animated:` (page 23) method) or by user interaction. The movie player whose state has changed is available as the object associated with the notification.

Availability
Available in iOS 3.2 and later.

Declared in
`MPMoviePlayerController.h`

MPMoviePlayerIsAirPlayVideoActiveDidChangeNotification

Posted when a movie player has started or ended playing a movie via AirPlay. There is no `userInfo` dictionary.

To find out whether AirPlay playback started or stopped, query the `airPlayVideoActive` (page 12) property. The movie player whose state has changed is available as the object associated with the notification.

Availability
Available in iOS 5.0 and later.

Declared in
`MPMoviePlayerController.h`

MPMoviePlayerLoadStateDidChangeNotification

Posted when a movie player's network buffering state has changed. There is no `userInfo` dictionary.

To retrieve the current load state of a movie player, access its `loadState` (page 16) property. The movie player whose state has changed is available as the object associated with the notification.

2012-09-19 | Copyright © 2012 Apple Inc. All Rights Reserved.

Availability
Available in iOS 3.2 and later.

Declared in
`MPMoviePlayerController.h`

MPMoviePlayerNowPlayingMovieDidChangeNotification

Posted when the currently playing movie has changed. There is no `userInfo` dictionary.

To retrieve the URL for currently playing movie, access the movie player's `contentURL` (page 13) property. The movie player whose state has changed is available as the object associated with the notification.

Availability
Available in iOS 3.2 and later.

Declared in
`MPMoviePlayerController.h`

MPMoviePlayerPlaybackDidFinishNotification

Posted when a movie has finished playing. The `userInfo` dictionary of this notification contains the `MPMoviePlayerPlaybackDidFinishReasonUserInfoKey` (page 33) key, which indicates the reason that playback finished. This notification is also sent when playback fails because of an error.

The movie player whose state has changed is available as the object associated with the notification.

This notification is *not* sent when a movie is displaying in fullscreen mode and the user taps Done. The Done button pauses playback and causes the movie player to exit fullscreen mode. To detect this scenario, register for other notifications such as `MPMoviePlayerDidExitFullscreenNotification` (page 36).

Availability
Available in iOS 2.0 and later.

Declared in
`MPMoviePlayerController.h`

MPMoviePlayerPlaybackStateDidChangeNotification

Posted when a movie player's playback state has changed. There is no `userInfo` dictionary.

2012-09-19 | Copyright © 2012 Apple Inc. All Rights Reserved.

Playback state can change programmatically (see *MPMediaPlayback Protocol Reference*) or by user interaction. To retrieve the playback state of a movie player, access its `playbackState` (page 18) property. The movie player whose state has changed is available as the object associated with the notification.

Availability
Available in iOS 3.2 and later.

Declared in
`MPMoviePlayerController.h`

MPMoviePlayerScalingModeDidChangeNotification

Posted when the scaling mode of a movie player has changed. There is no `userInfo` dictionary.

Scaling mode can change programmatically or by user interaction. To set or retrieve the scaling mode of a movie player, access its `scalingMode` (page 20) property. The movie player whose state has changed is available as the object associated with the notification.

Availability
Available in iOS 2.0 and later.

Declared in
`MPMoviePlayerController.h`

MPMoviePlayerThumbnailImageRequestDidFinishNotification

Posted when a request to capture a thumbnail from a movie has finished whether the request succeeded or failed. Upon successful capture of a thumbnail, the `userInfo` dictionary contains values for the following keys:

- `MPMoviePlayerThumbnailImageKey` (page 32)
- `MPMoviePlayerThumbnailTimeKey` (page 32)

If the capture request finished with an error, the `userInfo` dictionary contains values for the following two keys:

- `MPMoviePlayerThumbnailErrorKey` (page 32)
- `MPMoviePlayerThumbnailTimeKey` (page 32)

The movie player whose state has changed is available as the object associated with the notification. The methods to use for capturing movie thumbnails are described in "Generating Thumbnail Images" (page 11).

2012-09-19 | Copyright © 2012 Apple Inc. All Rights Reserved.

Availability
Available in iOS 3.2 and later.

Declared in
`MPMoviePlayerController.h`

MPMoviePlayerWillEnterFullscreenNotification

Posted when a movie player is about to enter full-screen mode. The `userInfo` dictionary contains keys whose values describe the transition animation used to enter full-screen mode. See "Fullscreen Notification Keys" (page 32).

A movie player can enter full screen mode programmatically (see the `setFullscreen:animated:` (page 23) method) or by user interaction. The movie player whose state has changed is available as the object associated with the notification.

Availability
Available in iOS 3.2 and later.

Declared in
`MPMoviePlayerController.h`

MPMoviePlayerWillExitFullscreenNotification

Posted when a movie player is about to exit full-screen mode. The `userInfo` dictionary contains keys whose values describe the transition animation used to exit full-screen mode. See "Fullscreen Notification Keys" (page 32).

A movie player can exit full screen mode programmatically (see the `setFullscreen:animated:` (page 23) method) or by user interaction. The movie player whose state has changed is available as the object associated with the notification.

Availability
Available in iOS 3.2 and later.

Declared in
`MPMoviePlayerController.h`

2012-09-19 | Copyright © 2012 Apple Inc. All Rights Reserved.

MPMovieSourceTypeAvailableNotification

Posted when the source type of a movie was previously unknown and is newly available. There is no `userInfo` dictionary.

To retrieve the source type of the movie, access the movie player's `movieSourceType` (page 17) property. The movie player whose state has changed is available as the object associated with the notification.

Availability
Available in iOS 3.2 and later.

Declared in
`MPMoviePlayerController.h`

MPMoviePlayerReadyForDisplayDidChangeNotification

Posted when the ready for display state changes.

To check whether a movie player is ready for display, access its `readyForDisplay` (page 19) property. The movie player whose display state has changed is available as the object associated with the notification.

Availability
Available in iOS 6.0 and later.

Declared in
`MPMoviePlayerController.h`

2012-09-19 | Copyright © 2012 Apple Inc. All Rights Reserved.

Deprecated MPMoviePlayerController Methods

A method identified as deprecated has been superseded and may become unsupported in the future.

Available in iOS 2.0 through iOS 3.1

backgroundColor

The color of the background area behind the movie. (Available in iOS 2.0 through iOS 3.1. Get the view from the `backgroundView` (page 13) property and set its color directly.)

```
@property (nonatomic, retain) UIColor *backgroundColor
```

Discussion
You should avoid using this property. It is available only when you use the `initWithContentURL:` (page 22) method to initialize the movie player controller object.

The receiver fades to and from the background color when transitioning to and from playback. Whenever the movie does not fill the screen exactly, this color is used to fill the area between the movie's frame and the edges of the screen.

The default color for this property is black. You can change this to other colors (including clear) to provide a more appropriate transition from your app's content to the movie content.

Availability
Available in iOS 2.0 through iOS 3.1.

Declared in
`MPMoviePlayerController.h`

movieControlMode

The user controls to display. (Available in iOS 2.0 through iOS 3.1. Use the `controlStyle` (page 13) property instead.)

```
@property (nonatomic) MPMovieControlMode movieControlMode
```

2012-09-19 | Copyright © 2012 Apple Inc. All Rights Reserved.

Discussion
Avoid using this property. It is available only when you use the `initWithContentURL:` (page 22) method to initialize the movie player controller object.

Determines the control (if any) the user has over movie playback. Different modes give the user access to different sets of playback controls, some of which allow the user to pause and resume playback and some of which do not.

This property is set to `MPMovieControlModeDefault` by default. See the "MPMovieControlMode" (page 33) enumeration for the available control modes.

Availability
Available in iOS 2.0 through iOS 3.1.

Declared in
`MPMoviePlayerController.h`

Deprecated in iOS 6.0

useApplicationAudioSession

A Boolean value that indicates whether the movie player should use the app's audio session. (Deprecated in iOS 6.0. There is not replacement for this property and its use is discouraged.)

```
@property (nonatomic) BOOL useApplicationAudioSession
```

Discussion
The default value of this property is `YES`. Setting this property to `NO` causes the movie player to use a system-supplied audio session with a nonmixable playback category.

> **Important:** In iOS 3.1 and earlier, a movie player always uses a system-supplied audio session. To obtain that same behavior in iOS 3.2 and newer, you must set this property's value to `NO`.

When this property is `YES`, the movie player shares the app's audio session. This give you control over how the movie player content interacts with your audio and with audio from other apps, such as the iPod. For important guidance on using this feature, see "Working with Movies and iPod Music" in *Audio Session Programming Guide*.

Changing the value of this property does not affect the currently playing movie. For the new setting to take effect, you must stop playback and then start it again.

2012-09-19 | Copyright © 2012 Apple Inc. All Rights Reserved.

Availability

Available in iOS 3.2 and later.

Deprecated in iOS 6.0.

Related Sample Code
MoviePlayer

Declared in

`MPMoviePlayerController.h`

Document Revision History

This table describes the changes to *MPMoviePlayerController Class Reference*.

Date	Notes
2012-09-19	Added new information for iOS 6.0.
2012-07-17	Removed line about not being allowed to pass nil in initWithContentURL:.
2011-11-17	Starting in iOS 5.0, a new movie player is not automatically prepared to play. See the discussion for the `initWithContentURL:` (page 22) instance method.
2011-10-12	Added descriptions for the `airPlayVideoActive` (page 12) property and the `MPMoviePlayerIsAirPlayVideoActiveDidChangeNotification` (page 36) notification.
2011-02-17	Added a description for the `allowsAirPlay` (page 12) property. Added descriptions for the `accessLog` (page 11) and `errorLog` (page 15) properties.
2010-07-13	Added information on playing streams whose URLs require access credentials. Added a code snippet to the MPMoviePlayerController (page 5) Overview to demonstrate how to correctly add a movie player to an application's view hierarchy.
2010-04-10	Added a description of the `timedMetadata` (page 24) method, which you can use to obtain time-based metadata from a streamed movie. Clarified the description of the `useApplicationAudioSession` (page 42) property.

2012-09-19 | Copyright © 2012 Apple Inc. All Rights Reserved.

Date	Notes
2010-03-23	Updated for iOS 3.2.
2009-08-17	Improved MPMoviePlayerController (page 5) introduction to describe how to play a series of movies. Improved Discussion section for `initWithContentURL:` (page 22) instance method to describe how to check for errors.
2009-06-02	Updated for iOS 3.0 Added description for `initialPlaybackTime` (page 16) property. Improved description for `scalingMode` (page 20) and `movieControlMode` (page 41) properties. Updated "Supported Formats" (page 8) section.
2008-05-27	New document that describes the class used to implement a full-screen movie player.

2012-09-19 | Copyright © 2012 Apple Inc. All Rights Reserved.

Apple Inc.
Copyright © 2012 Apple Inc.
All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, mechanical, electronic, photocopying, recording, or otherwise, without prior written permission of Apple Inc., with the following exceptions: Any person is hereby authorized to store documentation on a single computer for personal use only and to print copies of documentation for personal use provided that the documentation contains Apple's copyright notice.

No licenses, express or implied, are granted with respect to any of the technology described in this document. Apple retains all intellectual property rights associated with the technology described in this document. This document is intended to assist application developers to develop applications only for Apple-labeled computers.

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010

Apple, the Apple logo, AirPlay, Apple TV, iPod, and QuickTime are trademarks of Apple Inc., registered in the U.S. and other countries.

Times is a registered trademark of Heidelberger Druckmaschinen AG, available from Linotype Library GmbH.

iOS is a trademark or registered trademark of Cisco in the U.S. and other countries and is used under license.

Even though Apple has reviewed this document, APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS DOCUMENT IS PROVIDED "AS IS," AND YOU, THE READER, ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY.

IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT OR INACCURACY IN THIS DOCUMENT, even if advised of the possibility of such damages.

THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESS OR IMPLIED. No Apple dealer, agent, or employee is authorized to make any modification, extension, or addition to this warranty.

Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.