QTMovie Class Reference

QuickTime > Cocoa

2009-01-07

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

Simultaneously published in the United States and Canada.

Contents

QTMovie Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSCopying NSObject (NSObject)
Framework	/System/Library/Frameworks/QTKit.framework
Availability	Available in Mac OS X v10.4 and later.
Declared in	QTMovie.h
Related sample code	QTAudioExtractionPanel QTKitCreateMovie QTKitPlayer QTKitTimeCode QTMetadataEditor

Overview

The `QTMovie` class represents both a QuickTime movie and a movie controller. A movie is a collection of playable and editable media content. It describes the sources and types of the media in that collection and their spatial and temporal organization. These collections may be used for presentation (such as playback on the screen) or for the organization of media for processing (such as composition and transcoding to a different compression type). The collection may be as simple as a single file that plays at its natural size for its intrinsic duration, or it may be very complex (with multiple sources of content, rich composition rules, interactivity, and a variety of contingencies).

Just as a QuickTime movie contains a set of tracks, each of which defines the type, the segments, and the ordering of the media data it presents, a QTMovie object is associated with instances of the QTTrack class. In turn, a QTTrack object is associated with a single QTMedia object.

A QTMovie object can be initialized from a file, from a resource specified by a URL, from a block of memory, from a pasteboard, or from an existing QuickTime movie.

Once a QTMovie object has been initialized, it will typically be used in combination with a QTMovieView for playback.

An exception, `QTMovieUneditableException`, is raised whenever the client attempts to directly or indirectly edit a QTMovie object that is not currently set as editable (for instance, by calling `appendSelectionFromMovie:` on an uneditable movie).

Tasks

Determining If a Movie Can Be Initialized

`+ canInitWithFile:` (page 14)
Returns `YES` if the contents of the specified file can be used to initialize a QTMovie object.

`+ canInitWithURL:` (page 15)
Returns `YES` if the contents of the specified URL can be used to initialize a QTMovie object.

`+ canInitWithPasteboard:` (page 15)
Returns `YES` if the contents of the specified pasteboard can be used to initialize a QTMovie object.

`+ canInitWithDataReference:` (page 14)
Returns `YES` if the specified data reference can be used to initialize a QTMovie object.

`- initWithPasteboard:error:` (page 36)
Initializes a QTMovie object with the contents of the pasteboard specified by *`pasteboard`*.

Getting a List of Supported File Types

`+ movieFileTypes:` (page 17)
Returns an array of file types that can be opened as QuickTime movies.

`+ movieTypesWithOptions:` (page 18)
Returns an array of UTIs that QuickTime can open.

`+ movieUnfilteredFileTypes` (page 18)
Returns an array of file types that can be used to initialize a QTMovie object.

`+ movieUnfilteredPasteboardTypes` (page 19)
Returns an array of pasteboard types that can be used to initialize a QTMovie object.

Creating a Movie

`+ movie` (page 16)
Creates an empty QTMovie object.

`+ movieNamed:error:` (page 18)
Creates a QTMovie object initialized with the data from the QuickTime movie of the specified name in the application's bundle.

`+ movieWithData:error:` (page 21)
Creates a QTMovie object initialized with the data specified by *`data`*.

`+ movieWithURL:error:` (page 23)
Creates a QTMovie object initialized with the data in the URL specified by *`url`*.

`+ movieWithPasteboard:error:` (page 22)
Creates a QTMovie object initialized with the contents of the pasteboard specified by *`pasteboard`*.

`+ movieWithFile:error:` (page 21)
Creates a QTMovie object initialized with the data in the file specified by the name *`fileName`*.

+ `movieWithDataReference:error:` (page 21)
  Creates a QTMovie object intitalized with the data specified by the data reference *dataReference*.
+ `movieWithQuickTimeMovie:disposeWhenDone:error:` (page 22)
  Creates a QTMovie object initialized with the data from an existing QuickTime movie *movie*.
+ `movieWithAttributes:error:` (page 19)
  Creates a QTMovie object initialized with the attributes specified in *attributes*.

Controlling Movie Playback

- `autoplay` (page 25)
  Sets a movie to start playing when a sufficient amount of media data is available.
- `play` (page 40)
  Plays the movie.
- `stop` (page 47)
  Stops the movie playing.
- `gotoBeginning` (page 30)
  Repositions the play position to the beginning of the movie.
- `gotoEnd` (page 31)
  Repositions the play position to the end of the movie.
- `gotoNextSelectionPoint` (page 31)
  Repositions the movie to the next selection point.
- `gotoPreviousSelectionPoint` (page 31)
  Repositions the movie to the previous selection point.
- `gotoPosterFrame` (page 31)
  Repositions the play position to the movie's poster time.
- `setCurrentTime:` (page 44)
  Sets the movie's current time setting to `time`.
- `stepForward` (page 47)
  Sets the movie forward a single frame.
- `stepBackward` (page 47)
  Sets the movie backward a single frame.

Managing Threaded Operations of Movie Objects

+ `enterQTKitOnThread` (page 15)
  Performs any QuickTime-specific initialization for the current (non-main) thread; must be paired with a call to `exitQTKitOnThread`.
+ `enterQTKitOnThreadDisablingThreadSafetyProtection` (page 16)
  Performs any QuickTime-specific initialization for the current (non-main) thread, allowing non-threadsafe components; must be paired with a call to `exitQTKitOnThread`.
+ `exitQTKitOnThread` (page 16)
  Performs any QuickTime-specific shut-down for the current (non-main) thread; must be paired with a call to `enterQTKitOnThread` or `enterQTKitOnThreadDisablingThreadSafetyProtection`.

- `attachToCurrentThread` (page 25)
  Attaches the receiver to the current thread; returns YES if successful, NO otherwise.
- `detachFromCurrentThread` (page 28)
  Detaches the receiver from the current thread; returns YES if successful, NO otherwise.

Initializing a QTMovie

- `initWithFile:error:` (page 35)
  Initializes a QTMovie object with the data in the file specified by the name *`fileName`*.
- `initWithURL:error:` (page 37)
  Initializes a QTMovie object with the data in the URL specified by *`url`*.
- `initWithData:error:` (page 35)
  Initializes a QTMovie object with the data specified by *`data`*.
- `initWithDataReference:error:` (page 35)
  Initializes a QTMovie object with the data reference setting specified by *`dataReference`*.
- `initWithMovie:timeRange:error:` (page 36)
  Initializes a QTMovie object with some or all of the data from an existing QTMovie object *`movie`*.
- `initWithQuickTimeMovie:disposeWhenDone:error:` (page 36)
  Initializes a QTMovie object with the data from an existing QuickTime movie *`movie`*.
- `initWithAttributes:error:` (page 33)
  Initializes a QTMovie object with the attributes specified in *`attributes`*.

Getting Information About a Movie and Its Chapters

- `hasChapters` (page 32)
  Returns YES if the receiver has chapters, NO otherwise.
- `chapterCount` (page 26)
  Returns the number of chapters in the receiver, or 0 if there are no chapters.
- `chapters` (page 27)
  Returns an NSArray containing information about the chapters in the receiver.
- `addChapters` (page 23)
  Adds chapters to the receiver using the information specified in the chapters array.
- `removeChapters` (page 42)
  Removes any existing chapters from the receiver.
- `startTimeOfChapter:` (page 46)
  Returns a QTTime structure that is the start time of the chapter having the specified 0-based index in the list of chapters.
- `chapterIndexForTime:` (page 27)
  Returns the 0-based index of the chapter that contains the specified movie time.

Inspecting Movie Properties

- `duration` (page 29)
  Returns the duration of a QTMovie object as a structure of type `QTTime`.
- `currentTime` (page 27)
  Returns the current time of a QTMovie object as a structure of type `QTTime`.
- `rate` (page 41)
  Returns the current rate of a QTMovie object.
- `volume` (page 49)
  Returns the movie's volume as a scalar value of type `float`.
- `muted` (page 40)
  Returns the movie's mute setting.
- `movieWithTimeRange:error:` (page 39)
  Returns a QTMovie object whose data is the data in the specified time range.
- `attributeForKey:` (page 25)
  Returns the current value of the movie attribute *`attributeKey`*.
- `movieAttributes` (page 39)
  Returns a dictionary containing the current values of all defined movie attributes.

Managing QTMovie Idling States

- `setIdling:` (page 44)
  Sets the movie to idle `YES` or not to idle `NO`.
- `isIdling` (page 38)
  Returns the current idling state of a QTMovie object.

Setting QTMovie Properties

- `setRate:` (page 45)
  Sets the movie's rate to *`rate`*.
- `setVolume:` (page 46)
  Sets the movie's volume to *`volume`*.
- `setMuted:` (page 45)
  Sets the movie's mute setting to *`mute`*.

Setting Movie Attributes

- `setAttribute:forKey:` (page 43)
  Set the movie attribute *`attributeKey`* to the value specified by the *`value`* parameter.
- `setMovieAttributes:` (page 45)
  Set the movie attributes using the key-value pairs specified in the dictionary *`attributes`*.

Supporting Aperture Modes

- `generateApertureModeDimensions` (page 30)
  Adds information to a QTMovie needed to support aperture modes for tracks created with applications and/or versions of QuickTime that did not support aperture mode dimensions.
- `removeApertureModeDimensions` (page 42)
  Removes aperture mode dimension information from a movie's tracks.

Getting and Setting Selection Times

- `selectionStart` (page 43)
  Returns the start time of the movie's current selection as a QTTime structure.
- `selectionEnd` (page 43)
  Returns the end point of the movie's current selection as a QTTime structure.
- `selectionDuration` (page 43)
  Returns the duration of the movie's current selection as a QTTime structure.
- `setSelection:` (page 46)
  Sets the movie's selection to *`selection`*.

Getting Movie Tracks

- `tracks` (page 47)
  Returns an array of QTTrack objects associated with the receiver.
- `tracksOfMediaType:` (page 48)
  Returns an array of tracks with the specified media type.

Getting Movie Images

- `posterImage` (page 40)
  Returns an NSImage for the poster frame of a QTMovie.
- `currentFrameImage` (page 27)
  Returns an NSImage for the frame at the current time in a QTMovie.
- `frameImageAtTime:` (page 29)
  Returns an NSImage for the frame at the time *`time`* in a QTMovie.
- `frameImageAtTime:withAttributes:error:` (page 29)
  Returns an NSImage*, CIImage*, CGImageRef, CVPixelBufferRef, or CVOpenGLTextureRef for the movie image at the specified time

Storing Movie Data

- `initToWritableDataReference:error:` (page 32)
  Creates a new storage container at the location specified by `dataReference` and returns a QTMovie object that has that container as its default data reference.

- `initToWritableFile:error:` (page 33)
  Useful for directly passing filenames and data objects. The QTMovie returned by this method is editable.
- `initToWritableData:error:` (page 32)
  Useful for directly passing filenames and data objects. The QTMovie returned by this method is editable.
- `movieFormatRepresentation` (page 39)
  Returns the movie's data in an NSData object.
- `writeToFile:withAttributes:` (page 49)
  Returns YES if the movie file was successfully created and NO otherwise.
- `writeToFile:withAttributes:error:` (page 49)
  Returns an NSError object if an error occurs and if errorPtr is non-NULL.

Editing a Movie

- `replaceSelectionWithSelectionFromMovie:` (page 42)
  Replaces the current selection in a QTMovie with the current selection in *`movie`*.
- `appendSelectionFromMovie:` (page 24)
  Appends to a QTMovie the current selection in *`movie`*.
- `insertSegmentOfMovie:timeRange:atTime:` (page 38)
  Inserts into a QTMovie at time *`time`* the selection in `movie` delimited by the time range *`range`*.
- `insertSegmentOfMovie:fromRange:scaledToRange:` (page 38)
  Inserts the specified segment from the movie into the receiver, scaled to the range `dstRange`.
- `insertEmptySegmentAt:` (page 37)
  inserts into a QTMovie an empty segment delimited by the range *`range`*.
- `deleteSegment:` (page 28)
  Deletes from a QTMovie the segment delimited by *`segment`*.
- `scaleSegment:newDuration:` (page 42)
  Scales the QTMovie segment delimited by the segment *`segment`* so that it will have the new duration *`newDuration`*.
- `addImage:forDuration:withAttributes:` (page 24)
  Adds an image for the specified duration to the receiver, using attributes specified in the attributes dictionary.

Saving a Movie

- `canUpdateMovieFile` (page 26)
  Indicates whether a movie file can be updated with changes made to the movie object.
- `updateMovieFile` (page 48)
  Updates the movie file of a QTMovie.

Getting QTMovie Primitives

- `quickTimeMovie` (page 40)
  Returns the QuickTime movie associated with a QTMovie object.
- `quickTimeMovieController` (page 41)
  Returns the QuickTime movie controller associated with a QTMovie object.

Getting and Setting QTMovie Delegates

- `delegate` (page 28)
  Returns the delegate of a QTMovie object.
- `setDelegate:` (page 44)
  Sets the movie's delegate to *`delegate`*.
- `externalMovie:` (page 50) *delegate method*
  This method is called, if implemented by a QTMovie delegate object, when an external movie needs to be found (usually for a wired action targeted at an external movie).
- `movieShouldTask:` (page 51) *delegate method*
  If a QTMovie object has a delegate and that delegate implements this method, that method will be called before QTKit performs the standard idle processing on a movie.
- `movie:shouldContinueOperation:withPhase:atPercent:withAttributes:` (page 51) *delegate method*
  If implemented, this method is called periodically during lengthy operations (such as exporting a movie).
- `movie:linkToURL:` (page 50) *delegate method*
  Called to handle the mcAction `mcActionLinkToURL`.

Class Methods

canInitWithDataReference:

Returns `YES` if the specified data reference can be used to initialize a QTMovie object.

```
+ (BOOL)canInitWithDataReference:(QTDataReference*)dataReference
```

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

canInitWithFile:

Returns `YES` if the contents of the specified file can be used to initialize a QTMovie object.

```
+ (BOOL)canInitWithFile:(NSString *)fileName
```

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
QTAudioExtractionPanel
QTKitAdvancedDocument
QTKitFrameStepper
QTKitImport
QTKitPlayer

Declared In
`QTMovie.h`

canInitWithPasteboard:

Returns `YES` if the contents of the specified pasteboard can be used to initialize a QTMovie object.

```
+ (BOOL)canInitWithPasteboard:(NSPasteboard *)pasteboard
```

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

canInitWithURL:

Returns `YES` if the contents of the specified URL can be used to initialize a QTMovie object.

```
+ (BOOL)canInitWithURL:(NSURL *)url
```

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

enterQTKitOnThread

Performs any QuickTime-specific initialization for the current (non-main) thread; must be paired with a call to `exitQTKitOnThread`.

```
+ (void)enterQTKitOnThread
```

Availability
Mac OS X v10.5 and later.

Declared In
`QTMovie.h`

enterQTKitOnThreadDisablingThreadSafetyProtection

Performs any QuickTime-specific initialization for the current (non-main) thread, allowing non-threadsafe components; must be paired with a call to `exitQTKitOnThread`.

```
+ (void)enterQTKitOnThreadDisablingThreadSafetyProtection
```

Availability
Mac OS X v10.5 and later.

Related Sample Code
QTKitThreadedExport

Declared In
`QTMovie.h`

exitQTKitOnThread

Performs any QuickTime-specific shut-down for the current (non-main) thread; must be paired with a call to `enterQTKitOnThread` or `enterQTKitOnThreadDisablingThreadSafetyProtection`.

```
+ (void)exitQTKitOnThread
```

Availability
Mac OS X v10.5 and later.

Related Sample Code
QTKitThreadedExport

Declared In
`QTMovie.h`

movie

Creates an empty QTMovie object.

```
+ (id)movie
```

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
QTAudioExtractionPanel
QTKitImport
QTKitMovieShuffler
QTKitPlayer

Declared In
`QTMovie.h`

movieFileTypes:

Returns an array of file types that can be opened as QuickTime movies.

```
+ (NSArray *)movieFileTypes:(QTMovieTypeOptions)types
```

Discussion
Passing zero as the options parameter returns an array of all the common file types that QuickTime can open in place on the current system. This array includes the file type `.mov` and `.mqv`, and any files types that can be opened using a movie importer that does not need to write data into a new file while performing the import. This array excludes any file types for still images and any file types that require an aggressive movie importer (for instance, the movie importer for text files). The following values can be used to include some or all of the file types that are normally excluded:

```
enum {
    QTIncludeStillImageTypes  =  1 << 0,
    QTIncludeTranslatableTypes =  1 << 1,
    QTIncludeAggressiveTypes =  1 << 2,
    QTIncludeCommonTypes = 0,
    QTIncludeAllTypes = 0xffff
} QTMovieFileTypeOptions;
```

Constants	Description
`QTIncludeStillImageTypes` Available in Mac OS X v10.3 and later.	This value adds to the array all file types for still images that can be opened using a graphics importer.
`QTIncludeTranslatableTypes` Available in Mac OS X v10.3 and later. Declared in `QTMovie.h`.	This value adds to the array all file types for files that can be opened using a movie importer but for which a new file must be created.
`QTIncludeAggressiveTypes` Available in Mac OS X v10.3 and later. Declared in `QTMovie.h`.	This value adds to the array all file types for files that can be opened using a movie importer but that are not commonly used in connection with movies (for instance, text or HTML files).
`QTIncludeCommonTypes` Available in Mac OS X v10.3 and later. Declared in `QTMovie.h`.	This value adds to the array all common file types that QuickTime can open in place on the current system.
`QTIncludeAllTypes` Available in Mac OS X v10.3 and later. Declared in `QTMovie.h`.	This value adds to the array all file types that QuickTime can open on the current system, using any available movie or graphics importer.

Related Sample Code
LiveVideoMixer2
LiveVideoMixer3

QTKitAdvancedDocument

Declared In
QTMovie.h

movieNamed:error:

Creates a QTMovie object initialized with the data from the QuickTime movie of the specified name in the application's bundle.

```
+ (id)movieNamed:(NSString *)name
    error:(NSError **)errorPtr
```

Discussion
If a QTMovie object cannot be created, an NSError object is returned in the location pointed to by *errorPtr*. Pass NIL if you do not want an NSError object returned.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
CALayerEssentials

Declared In
QTMovie.h

movieTypesWithOptions:

Returns an array of UTIs that QuickTime can open.

```
+ (NSArray *)movieTypesWithOptions:(QTMovieFileTypeOptions)types
```

Discussion
This method gets an array of NSString objects that specify the uniform type identifiers (UTIs) for types of files that QuickTime can open. The types parameter is interpreted just like the types parameter to `+ (NSArray *)movieFileTypes:(QTMovieFileTypeOptions)types`.

Availability
Available in Mac OS X v10.5 and later.

Declared In
QTMovie.h

movieUnfilteredFileTypes

Returns an array of file types that can be used to initialize a QTMovie object.

```
+ (NSArray *)movieUnfilteredFileTypes
```

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
QTCoreImage101
QTCoreVideo103
QTCoreVideo202
QTKitMovieFrameImage
QTKitMovieShuffler

Declared In
`QTMovie.h`

movieUnfilteredPasteboardTypes

Returns an array of pasteboard types that can be used to initialize a QTMovie object.

```
+ (NSArray *)movieUnfilteredPasteboardTypes
```

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

movieWithAttributes:error:

Creates a QTMovie object initialized with the attributes specified in *`attributes`*.

```
+ (id)movieWithAttributes:(NSDictionary *)attributes
    error:(NSError **)errorPtr
```

Discussion
If a QTMovie object cannot be created, an NSError object is returned in the location pointed to by *`errorPtr`*. Pass `NIL` if you do not want an NSError object returned.

A new QTMovie object is created using the specified attributes. There are three types of attributes that can be included in this dictionary:

- Attributes that specify the location of the movie data
- Attributes that specify how the movie is to be instantiated
- Attributes that specify playback characteristics of the movie or other properties of the QTMovie object

The following is a list of the keys that specify the location of the movie data; at least one of these must occur in the dictionary. If more than one occurs, the first one in the dictionary is used.

Attribute	Description
`QTMovieFileNameAttribute`	The file name string of a QTMovie object; the value for this key is of type `NSString`.
`QTMovieURLAttribute`	The URL of a QTMovie object; the value for this key is of type `NSURL`.

Attribute	Description
QTMovieDataReferenceAttribute	The data reference of a QTMovie object; the value for this key is of type QTDataReference.
QTMoviePasteboardAttribute	The pasteboard representation of a QTMovie object; the value for this key is of type NSPasteboard.
QTMovieDataAttribute	The data representation of a QTMovie object; the value for this key is of type NSData.

The following is a list of the keys that specify movie instantiation options; none of these keys is required. If a key is missing, the specified default value is used.

Attribute	Description
QTMovieFileOffset-Attribute	The file offset of a QTMovie. The value for this key is of type NSNumber, which is interpreted as a long long. The default is 0.
QTMovieResolveData-RefsAttribute	The resolved data reference of a QTMovie. The value for this key is of type NSNumber, which is interpreted as a BOOL. Default: YES. If NO, QTMovie makes no attempt to resolve any external data references in a movie file.
QTMovieAskUnresolved-DataRefsAttribute	The asked unresolved data reference setting of a QTMovie. The value for this key is of type NSNumber, which is interpreted as a BOOL. Default: YES. If YES, QTMovie may display a dialog box prompting the user to help resolve any unresolved external data references in a movie file.
QTMovieOpenAsync-OKAttribute	The allowed synchronization opening setting of a QTMovie. The value for this key is of type NSNumber, which is interpreted as a BOOL. Default: YES. If YES, the initialization method returns immediately with a non-nil QTMovie object; however, the movie data might not all be loaded yet, so you may need to check the movie load state before performing certain operations on the movie. If NO, the movie data is loaded synchronously; when the initialization method returns with a non-nil QTMovie object, its data is completely loaded.

The following is a list of the new keys that specify movie playback characteristics or other properties of the QTMovie object; most other existing movie attributes can be included as well.

Attribute	Description
QTMovieAutoAlternatesAttribute	The auto-alternate setting of a QTMovie object. The value for this key is of type NSNumber, interpreted as a BOOL.
QTMovieIsActiveAttribute	The active setting; the value for this key is of type NSNumber, interpreted as a BOOL.
QTMovieDelegateAttribute	The delegate for a QTMovie object. The value for this key is of type NSObject.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTMovie.h

movieWithData:error:

Creates a QTMovie object initialized with the data specified by *data*.

```
+ (id)movieWithData:(NSData *)data
    error:(NSError **)errorPtr
```

Discussion
If a QTMovie object cannot be created, an NSError object is returned in the location pointed to by *errorPtr*. Pass NIL if you do not want an NSError object returned.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
QTKitCreateMovie
QTKitFrameStepper
QTKitImport

Declared In
QTMovie.h

movieWithDataReference:error:

Creates a QTMovie object intitalized with the data specified by the data reference *dataReference*.

```
+ (id)movieWithDataReference:(QTDataReference *)dataReference
    error:(NSError **)errorPtr
```

Discussion
If a QTMovie object cannot be created, an NSError object is returned in the location pointed to by *errorPtr*. Pass NIL if you do not want an NSError object returned.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTMovie.h

movieWithFile:error:

Creates a QTMovie object initialized with the data in the file specified by the name *fileName*.

```
+ (id)movieWithFile:(NSString *)fileName
    error:(NSError **)errorPtr
```

Discussion
The *fileName* is assumed to be a full path name for a file.

If a QTMovie object cannot be created, an NSError object is returned in the location pointed to by *errorPtr*. Pass NIL if you do not want an NSError object returned.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
QTAudioExtractionPanel
QTKitCommandLine
QTKitMovieFrameImage
QTKitPlayer
SillyFrequencyLevels

Declared In
QTMovie.h

movieWithPasteboard:error:

Creates a QTMovie object initialized with the contents of the pasteboard specified by *pasteboard*.

```
+ (id)movieWithPasteboard:(NSPasteboard *)pasteboard
    error:(NSError **)errorPtr
```

Discussion
These contents can be a QuickTime movie (of type Movie), a file path, or data of type QTMoviePasteboardType.

If a QTMovie object cannot be created, an NSError object is returned in the location pointed to by *errorPtr*. Pass NIL if you do not want an NSError object returned.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTMovie.h

movieWithQuickTimeMovie:disposeWhenDone:error:

Creates a QTMovie object initialized with the data from an existing QuickTime movie *movie*.

```
+ (id)movieWithQuickTimeMovie:(Movie)movie
    disposeWhenDone:(BOOL)dispose
    error:(NSError **)errorPtr
```

Discussion
The dispose parameter (a BOOL) indicates whether the QTKit should call DisposeMovie on the specified movie when the QTMovie object is deallocated. Passing YES effectively transfers "ownership" of the Movie to the QTKit. (Note that most applications will probably want to pass YES; passing NO means that the application wants to call DisposeMovie itself, perhaps so that it can operate on a Movie after it has been disassociated with a QTMovie object.)

If a QTMovie object cannot be created, an NSError object is returned in the location pointed to by *errorPtr*. Pass NIL if you do not want an NSError object returned.

Note that command-line tools that pass NO for the *disposeWhenDone* parameter must make sure to release the active autorelease pool before calling DisposeMovie on the specified QuickTime movie. Failure to do this may result in a crash. Tools that need to call DisposeMovie before releasing the main autorelease pool can create another autorelease pool associated with the movie.

Availability
Available in Mac OS X v10.3 and later.
Not available to 64-bit applications.

Related Sample Code
QTKitCreateMovie

Declared In
QTMovie.h

movieWithURL:error:

Creates a QTMovie object initialized with the data in the URL specified by *url*.

```
+ (id)movieWithURL:(NSURL *)url
    error:(NSError **)errorPtr
```

Discussion
If a QTMovie object cannot be created, an NSError object is returned in the location pointed to by *errorPtr*. Pass NIL if you do not want an NSError object returned.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
QTAudioExtractionPanel
QTKitCreateMovie
QTKitFrameStepper
QTKitPlayer
QTMetadataEditor

Declared In
QTMovie.h

Instance Methods

addChapters

Adds chapters to the receiver using the information specified in the chapters array.

```
- (void)addChapters:(NSArray *)chapters
    withAttributes:(NSDictionary *)attributes
    error:(NSError **)errorPtr
```

Discussion
Each array element is an NSDictionary containing key-value pairs. Currently two keys are defined for this dictionary, `QTMovieChapterName` and `QTMovieChapterStartTime`. The value for the `QTMovieChapterName` key is an NSString object that is the chapter name. The value for the `QTMovieChapterStartTime` key is an NSValue object that wraps a `QTTime` structure that indicates the start time of the chapter. The receiving QTMovie object must be editable or an exception will be raised.

The attributes dictionary specifies additional attributes for the chapters. Currently only one key is recognized for this dictionary, `QTMovieChapterTargetTrackAttribute`, which specifies the QTTrack in the receiver that is the target of the chapters; if none is specified, this method uses first video track in movie. If no video track is in the movie, this method uses the first audio track in the movie. If no audio track is in the movie, this method uses the first track in the movie. If an error occurs and errorPtr is non-NULL, then an NSError object is returned in that location.

Availability
Mac OS X v10.5 and later.

addImage:forDuration:withAttributes:

Adds an image for the specified duration to the receiver, using attributes specified in the attributes dictionary.

```
- (void)addImage:(NSImage *)image
    forDuration:(QTTime)duration
    withAttributes:(NSDictionary *)attributes
```

Discussion
Keys in the dictionary can be `QTAddImageCodecType` to select a codec type and `QTAddImageCodecQuality` to select a quality. Qualities are expected to be specified as NSNumbers, using the codec values like `codecNormalQuality`. (See ImageCompression.h for the complete list.) The attributes dictionary can also contain a value for the `QTTrackTimeScaleAttribute` key, which is used as the time scale of the new track, should one need to be created. The default time scale for a new track is 600.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
WritableFileDemo

Declared In
`QTMovie.h`

appendSelectionFromMovie:

Appends to a QTMovie the current selection in *`movie`*.

```
- (void)appendSelectionFromMovie:(id)movie
```

Discussion
If the movie is not editable, this method raises an exception.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

attachToCurrentThread

Attaches the receiver to the current thread; returns YES if successful, NO otherwise.

```
- (BOOL)attachToCurrentThread
```

Availability
Mac OS X v10.5 and later.

Related Sample Code
QTKitThreadedExport

Declared In
`QTMovie.h`

attributeForKey:

Returns the current value of the movie attribute *attributeKey*.

```
- (id)attributeForKey:(NSString *)attributeKey
```

Discussion
A list of supported movie attributes and their acceptable values can be found in the "Constants" (page 52) section.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
QTCoreVideo201
QTKitAdvancedDocument
QTKitFrameStepper
QTKitMovieShuffler
QTKitTimeCode

Declared In
`QTMovie.h`

autoplay

Sets a movie to start playing when a sufficient amount of media data is available.

```
- (void)autoplay
```

Discussion
The autoplay method configures a QTMovie object to begin playing as soon as enough data is available that the playback can continue uninterrupted to the end of the movie. This is most useful for movies being loaded from a remote URL or from an extremely slow local device. For movies stored on most local devices, this method has the same effect as the `-[QTMovie play]` method.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`QTMovie.h`

canUpdateMovieFile

Indicates whether a movie file can be updated with changes made to the movie object.

```
- (BOOL)canUpdateMovieFile
```

Discussion
This method returns `NO` if any of the following conditions are true:

- The movie is not associated with a file.
- The movie is not savable (has `'nsav'` user data set to 1).
- The movie file is not writable.
- The movie file does not contain a movie atom (indicating that the movie was imported from a non-movie format).

Otherwise, the method returns `YES`.

Using this method, an application can check first to see if the movie file can be updated; if not, it can prompt the user for a new name and location of a file in which to save the updated movie.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

chapterCount

Returns the number of chapters in the receiver, or 0 if there are no chapters.

```
- (NSInteger)chapterCount
```

Availability
Mac OS X v10.5 and later.

Declared In
`QTMovie.h`

chapterIndexForTime:

Returns the 0-based index of the chapter that contains the specified movie time.

```
- (NSInteger)chapterIndexForTime:(QTTime)time
```

Availability
Mac OS X v10.5 and later.

Declared In
`QTMovie.h`

chapters

Returns an NSArray containing information about the chapters in the receiver.

```
- (NSArray *)chapters
```

Discussion
Each array element is an NSDictionary containing key-value pairs. Currently two keys are defined for this dictionary, `QTMovieChapterName` and `QTMovieChapterStartTime`. The value for the `QTMovieChapterName` key is an NSString object that is the chapter name. The value for the `QTMovieChapterStartTime` key is an NSValue object that wraps a `QTTime` structure that indicates the start time of the chapter.

Availability
Mac OS X v10.5 and later.

Declared In
`QTMovie.h`

currentFrameImage

Returns an NSImage for the frame at the current time in a QTMovie.

```
- (NSImage *)currentFrameImage
```

Availability
Available in Mac OS X v10.3 and later.

See Also
- `frameImageAtTime:` (page 29)
- `posterImage` (page 40)

Declared In
`QTMovie.h`

currentTime

Returns the current time of a QTMovie object as a structure of type `QTTime`.

```
- (QTTime)currentTime
```

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

delegate

Returns the delegate of a QTMovie object.

```
- (id)delegate
```

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

deleteSegment:

Deletes from a QTMovie the segment delimited by *`segment`*.

```
- (void)deleteSegment:(QTTimeRange)segment
```

Discussion
If the movie is not editable, this method raises an exception.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
QTKitCommandLine

Declared In
`QTMovie.h`

detachFromCurrentThread

Detaches the receiver from the current thread; returns YES if successful, NO otherwise.

```
- (BOOL)detachFromCurrentThread
```

Discussion
These methods allow applications to manage QTMovie objects on non-main threads. Before any QTKit operations can be performed on a secondary thread, either `enterQTKitOnThread` or `enterQTKitOnThreadDisablingThreadSafetyProtection` must be called, and `exitQTKitOnThread` must be called before exiting the thread. A QTMovie object can be migrated from one thread to another by first calling `detachFromCurrentThread` on the first thread and then `attachToCurrentThread` on the second thread.

Availability
Mac OS X v10.5 and later.

Related Sample Code
QTKitThreadedExport

Declared In
`QTMovie.h`

duration

Returns the duration of a QTMovie object as a structure of type `QTTime`.

```
- (QTTime)duration
```

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
QTKitCreateMovie
QTKitMovieShuffler
QTKitTimeCode

Declared In
`QTMovie.h`

frameImageAtTime:

Returns an NSImage for the frame at the time *`time`* in a QTMovie.

```
- (NSImage *)frameImageAtTime:(QTTime)time
```

Availability
Available in Mac OS X v10.3 and later.

See Also
- `currentFrameImage` (page 27)
- `posterImage` (page 40)

Declared In
`QTMovie.h`

frameImageAtTime:withAttributes:error:

Returns an NSImage*, CIImage*, CGImageRef, CVPixelBufferRef, or CVOpenGLTextureRef for the movie image at the specified time

```
- (void *)frameImageAtTime:(QTTime)time
    withAttributes:(NSDictionary *)attributes
    error:(NSError **)errorPtr
```

Discussion
if an error occurs and the desired type of image cannot be created, then this returns nil and sets errorPtr to an NSError * describing the error. The dictionary of attributes can contain these keys:

- `QTMovieFrameImageSize`
- `QTMovieFrameImageType`
- `QTMovieFrameImageRepresentationsType`
- `QTMovieFrameImageOpenGLContext`
- `QTMovieFrameImagePixelFormat`
- `QTMovieFrameImageInterlaced`
- `QTMovieFrameImageHighQuality`
- `QTMovieFrameImageSingleField`

Note: All images returned by this method are autoreleased objects and must be retained by the caller if they are to be accessed outside of the current run loop cycle.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`QTMovie.h`

generateApertureModeDimensions

Adds information to a QTMovie needed to support aperture modes for tracks created with applications and/or versions of QuickTime that did not support aperture mode dimensions.

```
- (void)generateApertureModeDimensions
```

Discussion
If the image descriptions in video tracks lack tags describing clean aperture and pixel aspect ratio information, the media data is scanned to see if the correct values can be divined and attached. Then the aperture mode dimensions are calculated and set. Afterwards, the `QTTrackHasApertureModeDimensionsAttribute` property will be set to `YES` for those tracks. Tracks that do not support aperture modes are not changed.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

gotoBeginning

Repositions the play position to the beginning of the movie.

```
- (void)gotoBeginning
```

Discussion
If the movie is playing, the movie continues playing from the new position.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

gotoEnd

Repositions the play position to the end of the movie.

```
- (void)gotoEnd
```

Discussion
If the movie is playing in one of the looping modes, the movie continues playing accordingly; otherwise, play stops.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

gotoNextSelectionPoint

Repositions the movie to the next selection point.

```
- (void)gotoNextSelectionPoint
```

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

gotoPosterFrame

Repositions the play position to the movie's poster time.

```
- (void)gotoPosterFrame
```

Discussion
If no poster time is defined, the movie jumps to the beginning. If the movie is playing, the movie continues playing from the new position.

gotoPreviousSelectionPoint

Repositions the movie to the previous selection point.

```
- (void)gotoPreviousSelectionPoint
```

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

hasChapters

Returns YES if the receiver has chapters, NO otherwise.

```
- (BOOL)hasChapters
```

Availability
Mac OS X v10.5 and later.

Declared In
`QTMovie.h`

initToWritableData:error:

Useful for directly passing filenames and data objects. The QTMovie returned by this method is editable.

```
- (id)initToWritableData:(NSMutableData *)data
    error:(NSError **)errorPtr
```

Discussion
These methods––`initToWritableDataReference:error:`, `initToWritableFile:error:` and `initToWritableData:error:`––create an empty, writable storage container to which media data can be added (for example, using the QTMovie `addImage` method). The methods return QTMovie objects associated with those containers.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`QTMovie.h`

initToWritableDataReference:error:

Creates a new storage container at the location specified by `dataReference` and returns a QTMovie object that has that container as its default data reference.

```
- (id)initToWritableDataReference:(QTDataReference *)dataReference
    error:(NSError **)errorPtr
```

Availability
Available in Mac OS X v10.5 and later.

Declared In
`QTMovie.h`

initToWritableFile:error:

Useful for directly passing filenames and data objects. The QTMovie returned by this method is editable.

```
- (id)initToWritableFile:(NSString *)filename
    error:(NSError **)errorPtr
```

Availability
Available in Mac OS X v10.5 and later.

Related Sample Code
QTKitCreateMovie
WritableFileDemo

Declared In
`QTMovie.h`

initWithAttributes:error:

Initializes a QTMovie object with the attributes specified in *attributes*.

```
- (id)initWithAttributes:(NSDictionary *)attributes
    error:(NSError **)errorPtr
```

Discussion
If a QTMovie object cannot be created, an NSError object is returned in the location pointed to by *errorPtr*. Pass `NIL` if you do not want an NSError object returned.

A new QTMovie object is created using the specified attributes. There are three types of attributes that can be included in this dictionary:

- Attributes that specify the location of the movie data
- Attributes that specify how the movie is to be instantiated
- Attributes that specify playback characteristics of the movie or other properties of the QTMovie object

The following is a list of the keys that specify the location of the movie data; at least one of these must occur in the dictionary. If more than one occurs, the first one in the dictionary is used.

Attribute	Description
`QTMovieFileNameAttribute`	The file name string of a QTMovie object; the value for this key is of type `NSString`.
`QTMovieURLAttribute`	The URL of a QTMovie object; the value for this key is of type `NSURL`.
`QTMovieDataReferenceAttribute`	The data reference of a QTMovie object; the value for this key is of type `QTDataReference`.
`QTMoviePasteboardAttribute`	The pasteboard of a QTMovie object; the value for this key is of type `NSPasteboard`.

Attribute	Description
`QTMovieDataAttribute`	The data of a QTMovie object; the value for this key is of type `NSData`.

The following is a list of the keys that specify movie instantiation options; none of these keys is required. If a key is missing, the specified default value is used.

Attribute	Description
`QTMovieFileOffsetAttribute`	The file offset of a QTMovie. The value for this key is of type `NSNumber`, which is interpreted as a `long long`. The default is 0.
`QTMovieResolveData-RefsAttribute`	The resolved data reference setting of a QTMovie. The value for this key is of type `NSNumber`, which is interpreted as a `BOOL`. Default: `YES`.
`QTMovieAskUnresolved-DataRefsAttribute`	The asked unresolved data reference of a QTMovie. The value for this key is of type `NSNumber`, which is interpreted as a `BOOL`. Default: `YES`.
`QTMovieOpenAsyncOKAttribute`	The opened synchronization of a QTMovie. The value for this key is of type `NSNumber`, which is interpreted as a `BOOL`. Default: `YES`.

The following is a list of the new keys that specify movie playback characteristics or other properties of the QTMovie object; most other existing movie attributes can be included as well.

Attribute	Description
`QTMovieAuto-AlternatesAttribute`	The auto-alternate of a QTMovie object. The value for this key is of type `NSNumber`, interpreted as a `BOOL`.
`QTMovieIsActiveAttribute`	The active setting; the value for this key is of type `NSNumber`, interpreted as a `BOOL`.
`QTMovieDontInteract-WithUserAttribute`	When set in a dictionary passed to `movieWithAttributes` or `initWithAttributes`, this prevents QuickTime from interacting with the user during movie initialization. The value for this key is of type `NSNumber`, interpreted as a `BOOL`.
`QTMovieDelegateAttribute`	The delegate for a QTMovie object. The value for this key is of type `NSObject`.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
QTKitAdvancedDocument

Declared In
`QTMovie.h`

initWithData:error:

Initializes a QTMovie object with the data specified by *data*.

```
- (id)initWithData:(NSData *)data
    error:(NSError **)errorPtr
```

Discussion
If a QTMovie object cannot be created, an NSError object is returned in the location pointed to by *errorPtr*. Pass NIL if you do not want an NSError object returned.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTMovie.h

initWithDataReference:error:

Initializes a QTMovie object with the data reference setting specified by *dataReference*.

```
- (id)initWithDataReference:(QTDataReference *)dataReference
    error:(NSError **)errorPtr
```

Discussion
If a QTMovie object cannot be created, an NSError object is returned in the location pointed to by *errorPtr*. Pass NIL if you do not want an NSError object returned.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTMovie.h

initWithFile:error:

Initializes a QTMovie object with the data in the file specified by the name *fileName*.

```
- (id)initWithFile:(NSString *)fileName
    error:(NSError **)errorPtr
```

Discussion
The *fileName* is assumed to be a full path name for a file. If a QTMovie object cannot be created, an NSError object is returned in the location pointed to by *errorPtr*. Pass NIL if you do not want an NSError object returned.

Note that alias files should not be passed into this method; the client application is responsible for resolving aliases before handing them to QTKit methods.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
QTCoreImage101

QTKitButtonTester
QTKitMovieShuffler
QTQuartzPlayer
ViewController

Declared In
QTMovie.h

initWithMovie:timeRange:error:

Initializes a QTMovie object with some or all of the data from an existing QTMovie object *movie*.

```
- (id)initWithMovie:(QTMovie *)movie
    timeRange:(QTTimeRange)range
    error:(NSError **)errorPtr
```

Discussion
The section of data used is delimited by the range *range*. If a QTMovie object cannot be created, an NSError object is returned in the location pointed to by *errorPtr*. Pass NIL if you do not want an NSError object returned.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTMovie.h

initWithPasteboard:error:

Initializes a QTMovie object with the contents of the pasteboard specified by *pasteboard*.

```
- (id)initWithPasteboard:(NSPasteboard *)pasteboard
    error:(NSError **)errorPtr
```

Discussion
These contents can be a QuickTime movie (of type Movie), a file path, or data of type QTMoviePasteBoardType. If a QTMovie object cannot be created, an NSError object is returned in the location pointed to by *errorPtr*. Pass NIL if you do not want an NSError object returned.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTMovie.h

initWithQuickTimeMovie:disposeWhenDone:error:

Initializes a QTMovie object with the data from an existing QuickTime movie *movie*.

```
- (id)initWithQuickTimeMovie:(Movie)movie
    disposeWhenDone:(BOOL)dispose
    error:(NSError **)errorPtr
```

Discussion
This is the designated initializer for the QTMovie class. The `dispose` parameter (a `BOOL`) indicates whether the QTKit should call `DisposeMovie` on the specified movie when the QTMovie object is deallocated. Passing `YES` effectively transfers "ownership" of the Movie to the QTKit. (Note that most applications will probably want to pass `YES`; passing `NO` means that the application wants to call `DisposeMovie` itself, perhaps so that it can operate on a Movie after it has been disassociated from a QTMovie object.)

If a QTMovie object cannot be created, an NSError object is returned in the location pointed to by *`errorPtr`*. Pass `NIL` if you do not want an NSError object returned.

Availability
Available in Mac OS X v10.3 and later.
Not available to 64-bit applications.

Declared In
`QTMovie.h`

initWithURL:error:

Initializes a QTMovie object with the data in the URL specified by *`url`*.

```
- (id)initWithURL:(NSURL *)url
    error:(NSError **)errorPtr
```

Discussion
If a QTMovie object cannot be created, an NSError object is returned in the location pointed to by *`errorPtr`*. Pass `NIL` if you do not want an NSError object returned.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
QTKitFrameStepper

Declared In
`QTMovie.h`

insertEmptySegmentAt:

inserts into a QTMovie an empty segment delimited by the range *`range`*.

```
- (void)insertEmptySegmentAt:(QTTimeRange)range
```

Discussion
If the movie is not editable, this method raises an exception.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

insertSegmentOfMovie:fromRange:scaledToRange:

Inserts the specified segment from the movie into the receiver, scaled to the range `dstRange`.

```
- (void)insertSegmentOfMovie:(QTMovie *)movie
    fromRange:(QTTimeRange)srcRange
    scaledToRange:(QTTimeRange)dstRange
```

Discussion
This is essentially an Add Scaled operation on a movie. If the movie is not editable, this method raises an exception.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

insertSegmentOfMovie:timeRange:atTime:

Inserts into a QTMovie at time *`time`* the selection in `movie` delimited by the time range *`range`*.

```
- (void)insertSegmentOfMovie:(QTMovie *)movie
    timeRange:(QTTimeRange)range
    atTime:(QTTime)time
```

Discussion
If the movie is not editable, this method raises an exception.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
QTKitMovieShuffler

Declared In
`QTMovie.h`

isIdling

Returns the current idling state of a QTMovie object.

```
- (BOOL)isIdling
```

Discussion
This method allows you to manage the idling state of a QTMovie object, that is, whether it is being tasked. Note that movies attached to a background thread should not be idled; if they are idled, unexpected behavior can result.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`QTMovie.h`

movieAttributes

Returns a dictionary containing the current values of all defined movie attributes.

```
- (NSDictionary *)movieAttributes
```

Discussion
A list of supported movie attributes and their acceptable values can be found in the "Constants" (page 52) section.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

movieFormatRepresentation

Returns the movie's data in an NSData object.

```
- (NSData *)movieFormatRepresentation
```

Availability
Available in Mac OS X v10.3 and later.

See Also
`- writeToFile:withAttributes:` (page 49)

Related Sample Code
QTMetadataEditor

Declared In
`QTMovie.h`

movieWithTimeRange:error:

Returns a QTMovie object whose data is the data in the specified time range.

```
- (id)movieWithTimeRange:(QTTimeRange)range
    error:(NSError **)errorPtr
```

Discussion
If a QTMovie object cannot be created, an NSError object is returned in the location pointed to by *errorPtr*. Pass `NIL` if you do not want an NSError object returned.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

muted

Returns the movie's mute setting.

```
- (BOOL)muted
```

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTMovie.h

play

Plays the movie.

```
- (void)play
```

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
TrackFormatDemo
VideoViewer

Declared In
QTMovie.h

posterImage

Returns an NSImage for the poster frame of a QTMovie.

```
- (NSImage *)posterImage
```

Availability
Available in Mac OS X v10.3 and later.

See Also
- currentFrameImage (page 27),
- frameImageAtTime: (page 29)

Related Sample Code
QTKitMovieShuffler

Declared In
QTMovie.h

quickTimeMovie

Returns the QuickTime movie associated with a QTMovie object.

```
- (Movie)quickTimeMovie
```

Availability
Available in Mac OS X v10.3 and later.
Not available to 64-bit applications.

See Also
- `quickTimeMovieController` (page 41)

Related Sample Code
QTCoreVideo103
QTCoreVideo201
QTCoreVideo202
QTKitTimeCode
VideoViewer

Declared In
`QTMovie.h`

quickTimeMovieController

Returns the QuickTime movie controller associated with a QTMovie object.

```
- (MovieController)quickTimeMovieController
```

Availability
Available in Mac OS X v10.3 and later.
Not available to 64-bit applications.

See Also
- `quickTimeMovie` (page 40)

Related Sample Code
QTKitMovieShuffler

Declared In
`QTMovie.h`

rate

Returns the current rate of a QTMovie object.

```
- (float)rate
```

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
QTKitMovieShuffler

Declared In
`QTMovie.h`

removeApertureModeDimensions

Removes aperture mode dimension information from a movie's tracks.

```
- (void)removeApertureModeDimensions
```

Discussion
This method does not attempt to modify sample descriptions, so it may not completely reverse the effects of `generateApertureModeDimensions`. It sets the `QTMovieHasApertureModeDimensionsAttribute` property to `NO`.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

removeChapters

Removes any existing chapters from the receiver.

```
- (BOOL)removeChapters
```

Discussion
Returns YES if either the receiver had no chapters or the chapters were successfully removed from the receiver. Returns NO if the chapters could not for some reason be removed from the receiver. The receiving QTMovie object must be editable or an exception will be raised.

Availability
Mac OS X v10.5 and later.

Declared In
`QTMovie.h`

replaceSelectionWithSelectionFromMovie:

Replaces the current selection in a QTMovie with the current selection in *`movie`*.

```
- (void)replaceSelectionWithSelectionFromMovie:(id)movie
```

Discussion
If the movie is not editable, this method raises an exception.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

scaleSegment:newDuration:

Scales the QTMovie segment delimited by the segment *`segment`* so that it will have the new duration *`newDuration`*.

```
- (void)scaleSegment:(QTTimeRange)segment
    newDuration:(QTTime)newDuration
```

Discussion
If the movie is not editable, this method raises an exception.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

selectionDuration

Returns the duration of the movie's current selection as a QTTime structure.

```
- (QTTime)selectionDuration
```

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

selectionEnd

Returns the end point of the movie's current selection as a QTTime structure.

```
- (QTTime)selectionEnd
```

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

selectionStart

Returns the start time of the movie's current selection as a QTTime structure.

```
- (QTTime)selectionStart
```

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

setAttribute:forKey:

Set the movie attribute *attributeKey* to the value specified by the *value* parameter.

```
- (void)setAttribute:(id)value
    forKey:(NS String *)attributeKey
```

Discussion
A list of supported movie attributes and their acceptable values can be found in the "Constants" (page 52) section.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
QTCoreVideo103
QTCoreVideo202
QTKitCommandLine
QTKitMovieShuffler
ViewController

Declared In
`QTMovie.h`

setCurrentTime:

Sets the movie's current time setting to `time`.

```
- (void)setCurrentTime:(QTTime)time
```

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

setDelegate:

Sets the movie's delegate to *`delegate`*.

```
- (void)setDelegate:(id)delegate
```

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
QTKitProgressTester

Declared In
`QTMovie.h`

setIdling:

Sets the movie to idle `YES` or not to idle `NO`.

```
- (void)setIdling:(BOOL)state
```

Discussion
This method allows you to manage the idling state of a QTMovie object, that is, whether it is being tasked. Note that movies attached to a background thread should not be idled; if they are idled, unexpected behavior can result.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`QTMovie.h`

setMovieAttributes:

Set the movie attributes using the key-value pairs specified in the dictionary *attributes*.

```
- (void)setMovieAttributes:(NSDictionary *)attributes
```

Discussion
A list of supported movie attributes and their acceptable values can be found in the "Constants" (page 52) section.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

setMuted:

Sets the movie's mute setting to *mute*.

```
- (void)setMuted:(BOOL)mute
```

Discussion
Note that this does not affect the volume.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

setRate:

Sets the movie's rate to *rate*.

```
- (void)setRate:(float)rate
```

Discussion
For instance, 0.0 is stop, 1.0 is playback at normal speed, 2.0 is twice normal speed, and so on.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
QTCoreVideo102
QTCoreVideo103
QTCoreVideo201
QTCoreVideo202
QTCoreVideo301

Declared In
`QTMovie.h`

setSelection:

Sets the movie's selection to *`selection`*.

```
- (void)setSelection:(QTTimeRange)selection
```

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

setVolume:

Sets the movie's volume to *`volume`*.

```
- (void)setVolume:(float)volume
```

Discussion
Note that this does not affect the movie's stored settings.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

startTimeOfChapter:

Returns a QTTime structure that is the start time of the chapter having the specified 0-based index in the list of chapters.

```
- (QTTime)startTimeOfChapter:(NSInteger)chapterIndex
```

Availability
Mac OS X v10.5 and later.

Declared In
QTMovie.h

stepBackward

Sets the movie backward a single frame.

```
- (void)stepBackward
```

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTMovie.h

stepForward

Sets the movie forward a single frame.

```
- (void)stepForward
```

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTMovie.h

stop

Stops the movie playing.

```
- (void)stop
```

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
QTAudioExtractionPanel
QTKitMovieShuffler
QTKitPlayer

Declared In
QTMovie.h

tracks

Returns an array of QTTrack objects associated with the receiver.

```
- (NSArray *)tracks
```

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
QTMetadataEditor
TrackFormatDemo

Declared In
`QTMovie.h`

tracksOfMediaType:

Returns an array of tracks with the specified media type.

```
- (NSArray *)tracksOfMediaType:(NSString *)type
```

Discussion
The type parameter should be one of the media types defined by constants in `QTMedia.h` beginning with "QTMediaType", for instance, `QTMediaTypeVideo`.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
QTKitTimeCode

Declared In
`QTMovie.h`

updateMovieFile

Updates the movie file of a QTMovie.

```
- (BOOL)updateMovieFile
```

Discussion
Returns `YES` if the update succeeds and `NO` otherwise.

Availability
Available in Mac OS X v10.3 and later.

Related Sample Code
QTKitCommandLine
QTMetadataEditor
WritableFileDemo

Declared In
`QTMovie.h`

volume

Returns the movie's volume as a scalar value of type `float`.

```
- (float)volume
```

Discussion
The valid range is 0.0 to 1.0.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

writeToFile:withAttributes:

Returns YES if the movie file was successfully created and NO otherwise.

```
- (BOOL)writeToFile:(NSString *)fileNamewithAttributes
    :(NSDictionary *)attributes
```

Discussion
This method returns YES if the movie file was successfully created and NO otherwise. NO will also be returned if the load state of the target is less than `QTMovieLoadStateComplete`, in which case no attempt is made to write the QTMovie into a file. If the dictionary *`attributes`* contains an object whose key is `QTMovieFlatten`, then the movie is flattened into the specified file. If the dictionary *`attributes`* contains an object whose key is `QTMovieExport`, then the movie is exported into the specified file using a movie exporter whose type is specified by the value of the key `QTMovieExportType`. The value associated with the `QTMovieExportSettings` key should be an object of type NSData that contains an atom container of movie export settings.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- movieFormatRepresentation` (page 39)

Related Sample Code
QTKitCommandLine
QTKitMovieShuffler
QTKitProgressTester
QTKitThreadedExport

Declared In
`QTMovie.h`

writeToFile:withAttributes:error:

Returns an NSError object if an error occurs and if errorPtr is non-NULL.

```
- (BOOL)writeToFile:(NSString *)fileName
    withAttributes:(NSDictionary *)attributes
    error:(NSError **)errorPtr
```

Discussion
The method operates exactly like the existing `QTMovie writeToFile:withAttributes` method.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- movieFormatRepresentation` (page 39)

Declared In
`QTMovie.h`

Delegate Methods

externalMovie:

This method is called, if implemented by a QTMovie delegate object, when an external movie needs to be found (usually for a wired action targeted at an external movie).

```
- (QTMovie *)externalMovie:(NSDictionary *)dictionary
```

Discussion
The keys for the dictionary in this delegate method are: *`QTMovieTargetIDNotificationParameter`* and *`QTMovieTargetNameNotificationParameter`*. The *`QTMovieTargetIDNotificationParameter`* key indicates that the delegate should return a QTMovie object that has the specified movie ID. The *`QTMovieTargetNameNotificationParameter`* key indicates that the delegate should return a QTMovie object that has the specified movie name.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

movie:linkToURL:

Called to handle the mcAction `mcActionLinkToURL`.

```
- (BOOL)movie:(QTMovie *)movielinkToURL
    :(NSURL *)url
```

Discussion
Most applications will not need to install a delegate to handle this.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTMovie.h

movie:shouldContinueOperation:withPhase:atPercent:withAttributes:

If implemented, this method is called periodically during lengthy operations (such as exporting a movie).

```
- (BOOL)movie:(QTMovie *)movieshouldContinueOperation
    :(NSString *)opwithPhase
    :(QTMovieOperationPhase)phaseatPercent
    :(NSNumber *)percentwithAttributes
    :(NSDictionary *)attributes
```

Discussion
A delegate can implement this method. The `op` string is a localized string that indicates what the operation is. The `phase` indicates whether the operation is just beginning, ending, or is at a certain percentage of completion. If the phase is `QTMovieOperationUpdatePercentPhase`, then the `percent` parameter indicates the percentage of the operation completed. The `attributes` dictionary may be `NIL`; if not `NIL`, it is the same dictionary passed to a QTMovie method that caused the lengthy operation (for example, the `attributes` dictionary passed to `writeToFile`). The constants for this method are defined as follows:

```
typedef enum {
    QTMovieOperationBeginPhase = movieProgressOpen,
    QTMovieOperationUpdatePercentPhase = movieProgressUpdatePercent,
    QTMovieOperationEndPhase = movieProgressClose
}
```

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTMovie.h

movieShouldTask:

If a QTMovie object has a delegate and that delegate implements this method, that method will be called before QTKit performs the standard idle processing on a movie.

```
- (BOOL)movieShouldTask:(id)movie
```

Discussion
The delegate can cancel that normal processing by returning `YES`.

Availability
Available in Mac OS X v10.3 and later.

Declared In
QTMovie.h

Constants

The following constants specify the movie attributes that you can get and set using the `movieAttributes` and `setMovieAttributes` methods. To get or set a single attribute, use `attributeForKey` or `setAttribute`.

Constant	Description
`QTMovieActiveSegmentAttribute`	The active segment of a QTMovie object; the value for this key is of type `NSValue`, interpreted as a `QTTimeRange` structure. (**Deprecated.** This constant is available in Mac OS X 10.4 and later, but deprecated in Mac OS X 10.5.)
`QTMovieApertureModeAttribute`	Sets the aperture mode attribute on a QTMovie object to indicate whether aspect ratio and clean aperture correction should be performed. When a movie is in clean, production, or encoded pixels aperture mode, each track's dimensions are overridden by special dimensions for that mode. The original track dimensions are preserved and can be restored by setting the movie into classic aperture mode. Aperture modes are not saved in movies. The associated value is of type `NSString` and is assumed to be one of the following strings: `QTMovieApertureModeClassic`. No aspect ratio or clean aperture correction is performed. This is the default aperture mode and provides compatibility with behavior in QuickTime 7.0.x and earlier. If you call `-[QTTrack setDimensions]`, the movie is automatically switched to classic mode. `QTMovieApertureModeClean`. An aperture mode for general display. Where possible, video will be displayed at the correct pixel aspect ratio, trimmed to the clean aperture. A movie in clean aperture mode sets each track's dimensions to match the size returned by `-[QTTrack apertureModeDimensionsForMode:QTMovieApertureModeClean]`. `QTMovieApertureModeProduction`. `QTMovieApertureModeProduction`. An aperture mode for modal use in authoring applications. Where possible, video will be displayed at the correct pixel aspect ratio, but without trimming to the clean aperture so that the edge processing region can be viewed. A movie in production aperture mode sets each track's dimensions to match the size returned by `-[QTTrack apertureModeDimensionsForMode:QTMovieApertureModeProduction]`. `QTMovieApertureModeEncodedPixels`. `QTMovieApertureModeEncodedPixels`. An aperture mode for technical use. Displays all encoded pixels with no aspect ratio or clean aperture compensation. A movie in encoded pixels aperture mode sets each track's dimensions to match the size returned by `-[QTTrack apertureModeDimensionsForMode:QTMovieApertureModeEncodedPixels]`.
`QTMovieAutoAlternatesAttribute`	The auto-alternate state of a QTMovie object. The value for this key is of type `NSNumber`, interpreted as a `BOOL`.
`QTMovieCopyrightAttribute`	The copyright string of a QTMovie object; the value for this key is of type `NSString`.
`QTMovieCreationTimeAttribute`	The creation time of a QTMovie object; the value for this key is of type `NSDate`.

Constant	Description
`QTMovieCurrentSizeAttribute`	The current size of a QTMovie object; the value for this key is of type `NSValue`, interpreted as an `NSSize` structure.
`QTMovieCurrentTimeAttribute`	The current time of a QTMovie object; the value for this key is of type `NSValue`, interpreted as a QTTime structure.
`QTMovieDataSizeAttribute`	The data size of a QTMovie. The value for this key is of type `NSNumber`, which is interpreted as a `long long`.
`QTMovieDelegateAttribute`	The delegate for a QTMovie object. The value for this key is of type `NSObject`.
`QTMovieDisplayNameAttribute`	The display name of a QTMovie object. A display name is stored as user data in a movie file and hence may differ from the base name of the movie's filename or URL. The value for this key is of type `NSString`.
`QTMovieDontInteractWithUserAttribute`	When set in a dictionary passed to `movieWithAttributes` or `initWithAttributes`, this prevents QuickTime from interacting with the user during movie initialization. The value for this key is of type `NSNumber`, interpreted as a `BOOL`.
`QTMovieDurationAttribute`	The duration of a QTMovie object; the value for this key is of type `NSValue`, interpreted as a `QTTime` structure.
`QTMovieEditableAttribute`	The editable setting; the value for this key is of type `NSNumber`, interpreted as a `BOOL`. This value is `YES` if the movie can be edited.
`QTMovieFileNameAttribute`	The file name string of a QTMovie object; the value for this key is of type `NSString`.
`QTMovieHasApertureModeDimensionsAttribute`	The aperture mode dimensions set on any track in this QTMovie object, even if those dimemsions are all identical to the classic dimensions (as is the case for content with square pixels and no edge-processing region). The value for this key is of type `NSNumber`, interpreted as a `BOOL`.
`QTMovieHasAudioAttribute`	The audio data setting; the value for this key is of type `NSNumber`, interpreted as a `BOOL`. This value is `YES` if the movie contains audio data.
`QTMovieHasDurationAttribute`	The duration setting; the value for this key is of type `NSNumber`, interpreted as a `BOOL`. This value is `YES` if the movie has a duration. (Some types of movies, for instance QuickTime VR movies, have no duration.)
`QTMovieHasVideoAttribute`	The video data setting; the value for this key is of type `NSNumber`, interpreted as a `BOOL`. This value is `YES` if the movie contains video data.
`QTMovieIsActiveAttribute`	The active setting; the value for this key is of type `NSNumber`, interpreted as a `BOOL`.
`QTMovieIsInteractiveAttribute`	The interactive setting; the value for this key is of type `NSNumber`, interpreted as a `BOOL`. This value is `YES` if the movie is interactive.
`QTMovieIsLinearAttribute`	The linear setting; the value for this key is of type `NSNumber`, interpreted as a `BOOL`. This value is `YES` if the movie is linear, as opposed to a non-linear QuickTime VR movie.

Constant	Description
`QTMovieIsSteppable-Attribute`	The steppable setting; the value for this key is of type `NSNumber`, interpreted as a `BOOL`. This value is `YES` if the movie can step from frame to frame.

`QTMovieLoadState-Attribute`	The load state value; the value for this key is of type `NSNumber`, interpreted as a `long`.
	`enum {`
	`  QTMovieLoadStateError = -1L,  // an error occurred while loading the movie`
	`  QTMovieLoadStateLoading = 1000, // the movie is loading`
	`  QTMovieLoadStateLoaded = 2000, // the movie atom has loaded; it's safe to query movie properties`
	`  QTMovieLoadStatePlayable = 10000, // the movie has loaded enough media data to begin playing`
	`  QTMovieLoadStatePlaythroughOK = 20000, // the movie has loaded enough media data to play through to the end`
	`  QTMovieLoadStateComplete = 100000L // the movie has loaded completely`
	`};`
	
	The `attributeForKey: QTMovieLoadStateAttribute` returns an NSNumber that wraps a long integer; the enumerated constants shown above are the possible values of that long integer.
	Mac OS X v10.5 and later.

`QTMovieLoops-Attribute`	The looping setting; the value for this key is of type `NSNumber`, interpreted as a `BOOL`. This value is `YES` if the movie is set to loop.
`QTMovieLoopsBackAnd-ForthAttribute`	The palindrome looping setting; the value for this key is of type `NSNumber`, interpreted as a `BOOL`. This value is `YES` if the movie is set to loop back and forth. Note that `QTMovieLoopsAttribute` and `QTMovieLoopsBackAndForthAttribute` are independent and indeed exclusive. `QTMovieLoopsAttribute` is used to get and set the state of normal looping; `QTMovieLoopsBackAndForthAttribute` is used to get and set the state of palindrome looping.
`QTMovieModification-TimeAttribute`	The modification time of a QTMovie object; the value for this key is of type `NSDate`.
`QTMovieMuted-Attribute`	The mute setting; the value for this key is of type `NSNumber`, interpreted as a `BOOL`. This value is `YES` if the movie volume is muted.
`QTMovieNatural-SizeAttribute`	The natural size of a QTMovie object; the value for this key is of type `NSValue`, interpreted as an `NSSize` structure.
`QTMoviePlaysAll-FramesAttribute`	The play-all-frames setting; the value for this key is of type `NSNumber`, interpreted as a `BOOL`. This value is `YES` if the movie will play all frames.
`QTMoviePlays-SelectionOnly-Attribute`	The play-selection setting; the value for this key is of type `NSNumber`, interpreted as a `BOOL`. This value is `YES` if the movie will play only the current selection.
`QTMoviePosterTime-Attribute`	The movie poster time of a QTMovie object; the value for this key is of type `NSValue`, interpreted as a `QTTime` structure.

Constant	Description
`QTMoviePreferredMutedAttribute`	The preferred mute setting; the value for this key is of type `NSNumber`, interpreted as a `BOOL`. This value is `YES` if the movie preferred mute setting is muted.
`QTMoviePreferredRateAttribute`	The preferred rate; the value for this key is of type `NSNumber`, interpreted as a `float`.
`QTMoviePreferredVolumeAttribute`	The preferred volume; the value for this key is of type `NSNumber`, interpreted as a `float`.
`QTMoviePreviewModeAttribute`	The preview mode setting; the value for this key is of type `NSNumber`, interpreted as a `BOOL`. This value is `YES` if the movie is in preview mode.
`QTMoviePreviewRangeAttribute`	The preview range of a QTMovie object; the value for this key is of type `NSValue`, interpreted as a `QTTimeRange` structure.
`QTMovieRateAttribute`	The movie rate; the value for this key is of type `NSNumber`, interpreted as a `float`.
`QTMovieRateChangesPreservePitchAttribute`	When the playback rate is not unity, audio must be resampled in order to play at the new rate. The default resampling affects the pitch of the audio (for example, playing at 2x speed raises the pitch by an octave, 1/2x lowers an octave). If this property is set on the Movie, an alternative algorithm is used, which alters the speed without changing the pitch. As this is more computationally expensive, this property may be silently ignored on some slow CPUs.
`QTMovieSelectionAttribute`	The selection range of a QTMovie object; the value for this key is of type `NSValue`, interpreted as a `QTTimeRange` structure.
`QTMovieTimeScaleAttribute`	The movie time scale; the value for this key is of type `NSNumber`, interpreted as a `long`. In Mac OS X 10.5 and later, this attribute is gettable and settable. In general, you should set this attribute only on newly-created movies or on movies that have not been edited. Also, you should only increase the time scale value, and you should try to use integer multiples of the existing time scale. In earlier versions of Mac OS X, this attribute is gettable only.
`QTMovieURLAttribute`	The URL of a QTMovie object; the value for this key is of type `NSURL`.
`QTMovieVolumeAttribute`	The movie volume; the value for this key is of type `NSNumber`, interpreted as a `float`.

The following constants specify items in dictionaries passed to QTMovie notifications and delegate methods.

Constant	Description
`QTMovieMessageNotificationParameter`	Used as a key in the userInfo dictionary passed to the `QTMovieMessageNotification` notification to indicate the message. The associated value is an `NSString`.
`QTMovieRateDidChangeNotificationParameter`	Used as a key in the userInfo dictionary passed to the `QTMovieRateDidChangeNotification` notification to indicate the new playback rate. The associated value is an `NSNumber` that holds a `float`.

Constant	Description
`QTMovieStatusFlagsNotificationParameter`	Used as a key in the userInfo dictionary passed to the `QTMovieStatusStringPostedNotification` notification to indicate status flags. The associated value is an `NSNumber` that holds a `long`.
`QTMovieStatusCodeNotificationParameter`	Used as a key in the userInfo dictionary passed to the `QTMovieStatusStringPostedNotification` notification to indicate a status code (or error code). The associated value is an `NSNumber` that holds an `int`.
`QTMovieStatusStringNotificationParameter`	Used as a key in the userInfo dictionary passed to the `QTMovieStatusStringPostedNotification` notification to indicate a status string.
`QTMovieTargetIDNotificationParameter`	Used as a key in the dictionary passed to the `externalMovie:` delegate method to indicate that the delegate should return a QTMovie object that has the movie ID specified by the key's value.
`QTMovieTargetNameNotificationParameter`	Used as a key in the dictionary passed to the `externalMovie:` delegate method to indicate that the delegate should return a QTMovie object that has the movie name specified by the key's value.

The following constants are dictionary keys that you can use to specify movie attributes, using the `writeToFile` method.

Constant	Description
`QTMovieExport`	The movie export setting; the value for this key is of type `NSNumber`, interpreted as a `BOOL`.
`QTMovieExportType`	The movie export type; the value for this key is of type `NSNumber`, interpreted as a `long`.
`QTMovieFlatten`	The movie flatten setting; the value for this key is of type `NSNumber`, interpreted as a `BOOL`.
`QTMovieExportSettings`	Information to come.
`QTMovieExportManufacturer`	The export manufacturer value; the value for this key is of type `NSNumber`, interpreted as a `long`.

The following constants are dictionary keys that you can use to specify movie attributes, using the `addImage` method.

Constant	Description
`QTAddImageCodecType`	The image codec string; the value for this key is of type `NSString`.
`QTAddImageCodecQuality`	The image codec value; the value for this key is of type `NSNumber`.

The following is a dictionary of attributes can contain these keys, using the `frameImageAtTime:withAttributes:error:` method.

Constant	Description
`QTMovieFrameImageSize`	Size of the image. Value is an NSValue containing an NSSize record. The default image size is the current movie size.
`QTMovieFrameImageType`	Type of the image. Value is an NSString. The default image type is NSImage.
`QTMovieFrameImage-RepresentationsType`	For NSImage, the image representations in the image. Value is an NSArray of NSString; strings are, for example, NSBitmapImageRep class description. The default is NSBitmapImageRep.
`QTMovieFrameImage-OpenGLContext`	For CVOpenGLTextureRef, the OpenGL context to use. Value is an NSValue (CGLContextObj).
`QTMovieFrameImagePixelFormat`	For CVOpenGLTextureRef, the pixel format to use. Value is an NSValue (CGLPixelFormatObj).
`QTMovieFrameImageInterlaced`	Image is interlaced. Value is an NSNumber (BOOL) (default = NO).
`QTMovieFrameImageHighQuality`	Image is high quality. Value is an NSNumber (BOOL) (default = YES).
`QTMovieFrameImageSingleField`	Image is single field. Value is an NSNumber (BOOL) (default = YES). The returned object is an autorelease object.

The following constants are data locators that you can use to specify movie attributes, using the `movieWithAttributes` and `initWithAttributes` methods.

Constant	Description
`QTMovieDataReferenceAttribute`	The data reference of a QTMovie object.
`QTMoviePasteboardAttribute`	The pasteboard setting of a QTMovie object.
`QTMovieDataAttribute`	The data of a QTMovie object.

The following constants are movie instantiation options that you can use to specify movie attributes, using the `movieWithAttributes` and `initWithAttributes` methods.

Constant	Description
`QTMovieFileOffsetAttribute`	The file offset value; the value for this key is of type `NSNumber`, interpreted as a `long long`.
`QTMovieResolveDataRefAttribute`	The resolved data reference setting; the value for this key is of type `NSNumber`, interpreted as a `BOOL`.
`QTMovieAskUnresolved-DataRefAttribute`	The unresolved data reference setting; the value for this key is of type `NSNumber`, interpreted as a `BOOL`.

Constant	Description
`QTMovieOpenAsyncOKAttribute`	The open async setting; the value for this key is of type `NSNumber`, interpreted as a `BOOL`.

These constants allow applications to get information about a movie and its chapters, and to navigate within a movie by chapters. Since chapters are a reasonably common feature of movies and podcasts, QTKit enables developers to create them.

Constant	Description
`QTMovieChapterName`	A key indicating the chapter name in the dictionaries that are array elements in the array returned by `QTMovie chapters` or passed to QTMovie `addChapters: withAttributes:error`.
`QTMovieChapterStartTime`	Aey indicating the chapter start time in the dictionaries that are array elements in the array returned by `QTMovie chapters` or passed to `QTMovie addChapters: withAttributes:error`.
`QTMovieChapterTarget-TrackAttribute`	A key indicating the track in the `QTMovie object` that is the target of the chapter track.

Notifications

QTMovieApertureModeDidChangeNotification

Issued when the aperture mode of the target QTMovie object changes.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

QTMovieChapterDidChangeNotification

Issued when the chapter associated with QTMovie changes.

This notification contains no information in the userInfo dictionary.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

QTMovieChapterListDidChangeNotification

Issued when the chapter list associated with QTMovie changes.

This notification contains no information in the userInfo dictionary.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

QTMovieCloseWindowRequestNotification

Sent when a request is made to close the movie's window.

This notification contains no information in the userInfo dictionary.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

QTMovieDidEndNotification

Sent when the movie is "done" or at its end.

This notification contains no userInfo parameters. It is equivalent to the standard player controller's `mcActionMovieFinished` action.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

QTMovieEditabilityDidChangeNotification

Sent when the editable state of a movie has changed.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

QTMovieEditedNotification

Sent when a movie has been edited.

This notification contains no userInfo dictionary.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

QTMovieEnterFullScreenRequestNotification

Sent when a request is made to play back a movie in full screen mode.

This notification contains no information in the userInfo dictionary.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

QTMovieExitFullScreenRequestNotification

Sent when a request is made to play back a movie in normal windowed mode.

This notification contains no information in the userInfo dictionary.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

QTMovieLoadStateDidChangeNotification

Sent when the load state of a movie has changed.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

QTMovieLoopModeDidChangeNotification

Sent when a change is made in a movie's looping mode.

This notification contains no information in the userInfo dictionary.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

QTMovieMessageStringPostedNotification

Sent when a movie message has been received by the movie controller.

Movie messages can be sent to an application by wired actions (for instance, a wired sprite) or by code that issues the `mcActionShowMessageString` movie controller action. The userInfo dictionary contains a single entry whose value is of type `NSString`, which is the movie message.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

QTMovieRateDidChangeNotification

Sent when the rate of a movie has changed.

The userInfo dictionary contains a single entry whose value is of type `NSNumber` that represents a `float`, which is the new rate.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

QTMovieSelectionDidChangeNotification

Sent when the selection of a movie has changed.

This notification contains no userInfo dictionary.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

QTMovieSizeDidChangeNotification

Sent when the size of a movie has changed.

This notification contains no userInfo dictionary.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

QTMovieStatusStringPostedNotification

Status messages can be sent by QuickTime's streaming components or by any code that wants to display a message in the movie controller bar status area.

The userInfo dictionary contains a single entry whose value is of type `NSString`, which is the status message.

The following are keys (notification parameters) for userInfo items for the `QTMovieStatusStringPostedNotification` notification `QTMovieStatusCodeNotificationParameter` and `QTMovieStatusStringNotificationParameter`.

A status string notification can indicate an error (in which case `QTMovieStatusCodeNotificationParameter` will have a value), or it can contain a string (in which case `QTMovieStatusStringNotificationParameter` will have a value). For more information, see `mcActionShowStatusString`.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

QTMovieTimeDidChangeNotification

Sent when the time in a movie has changed to a value other than what it would be during normal playback.

The `QTMovieTimeDidChangeNotification` is fired whenever the movie time changes to a time other than what it would be during normal playback. So, for example, this notification is not fired every frame.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

QTMovieVolumeDidChangeNotification

Sent when the volume of a movie has changed.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`QTMovie.h`

Document Revision History

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

Date	Notes
2009-01-07	Corrected the description of QuickTime VR attribute. Minor typo fixes.
2007-10-31	Updated for Mac OS X v10.5. Fixed an error in the description of a QuickTime VR implementation.
2007-07-18	Updated for Mac OS X v10.5 with new methods.
2006-09-05	Updated information on new methods, notifications, and attributes available in QuickTime 7.1.

Document Revision History

Index

M

P

Q

R

S

T

U

V

W