Movie Manager Reference

QuickTime > Movie Basics

2006-12-14

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

Simultaneously published in the United States and Canada.

Contents

Movie Manager Reference

Framework:	Frameworks/QuickTime.framework
Declared in	Controls.h HIObject.h MacTypes.h Movies.h TextEdit.h

Overview

QuickTime movies have certain overall timing and other presentation characteristics that an application can manage, including the presentation of special kinds of media such as flash media and sprites.

Functions by Task

Controlling Movie Playback

`GoToBeginningOfMovie` (page 89)
Repositions a movie to play from its start.

`GoToEndOfMovie` (page 90)
Repositions a movie to play from its end.

`StartMovie` (page 181)
Starts the movie playing from the current movie time.

`StopMovie` (page 182)
Stops the playback of a movie.

Creating and Disposing of Time Bases

`ChooseMovieClock` (page 29)
Searches media handlers to find the best clock for a movie.

`DisposeTimeBase` (page 37)
Disposes of a time base once you are finished with it.

`GetTimeBaseMasterClock` (page 78)
Determines the clock component that is assigned to a time base.

`GetTimeBaseMasterTimeBase` (page 79)
Determines the master time base that is assigned to a time base.

`NewTimeBase` (page 110)
Obtains a new time base.

`QTGetWallClockTimeBase` (page 120)
Returns the system's real-time time base.

`SetMovieMasterClock` (page 139)
Assigns a clock component to a movie.

`SetMovieMasterTimeBase` (page 140)
Assigns a master time base to a movie.

`SetTimeBaseMasterClock` (page 153)
Assigns a clock component to a time base.

`SetTimeBaseMasterTimeBase` (page 154)
Assigns a master time base to a time base.

`SetTimeBaseZero` (page 158)
Changes the offset from a time base to either its master time base or its clock component.

Determining Movie Creation and Modification Time

`GetMovieCreationTime` (page 57)
Returns the movie's creation date and time information.

`GetMovieModificationTime` (page 62)
Returns a movie's modification date and time.

Disabling Movies and Tracks

`GetMovieActive` (page 53)
Determines whether a movie is currently active.

`SetMovieActive` (page 133)
Activates or deactivates a movie.

Enhancing Movie Playback Performance

`AbortPrePrerollMovie` (page 25)
Terminates the operation of PrePrerollMovie.

`GetMovieActiveSegment` (page 53)
Determines what portion of a movie is currently active for playing.

`LoadMediaIntoRam` (page 94)
Loads a media's data into memory.

`LoadMovieIntoRam` (page 95)
Loads a movie's data into memory.

`LoadTrackIntoRam` (page 96)
Loads a track's data into memory.

`PrePrerollMovie` (page 112)
Sets up any necessary network connections to receive streaming content.

`PrerollMovie` (page 113)
Prepares a portion of a movie for playback.

`SetMovieActiveSegment` (page 134)
Defines a movie's active segment.

Error Functions

`ClearMoviesStickyError` (page 30)
Clears the sticky error value.

`GetMoviesError` (page 70)
Returns the contents of the current error value and resets the current error value to 0.

`GetMoviesStickyError` (page 71)
Returns the contents of the sticky error value.

`SetMoviesErrorProc` (page 147)
Performs custom error notification.

Generating Pictures From Movies

`GetMoviePict` (page 63)
Creates a QuickDraw picture from a specified movie at a specified time.

`GetMoviePosterPict` (page 64)
Creates a QuickDraw picture that contains a movie's poster.

`GetTrackPict` (page 88)
Creates a QuickDraw picture from a specified track at a specified time.

High-Level Movie Editing Functions

`GetMovieSelection` (page 69)
Returns information about a movie's current selection.

`SetMovieSelection` (page 146)
Sets a movie's current selection.

Initializing the Movie Toolbox

`EnterMovies` (page 38)
Initializes the Movie Toolbox and creates a private storage area for your application.

`ExitMovies` (page 40)
Automatically called when an application quits.

Managing Movie Sprites

`SpriteMediaCountImages` (page 162)
Retrieves the number of images that currently exist in a sprite track.

`SpriteMediaCountSprites` (page 163)
Retrieves the number of sprites that currently exist in a sprite track.

`SpriteMediaGetDisplayedSampleNumber` (page 166)
Retrieves the number of the sprite media sample that is currently being displayed.

`SpriteMediaGetImageName` (page 166)
Returns the name of the image with the specified index from the current key frame sample.

`SpriteMediaGetIndImageDescription` (page 167)
Retrieves an image description for a specified image in a sprite track.

`SpriteMediaGetIndImageProperty` (page 168)
Returns a property value for a sprite image specified by an index.

`SpriteMediaGetSpriteName` (page 170)
Returns the name of the sprite with the specified ID from the currently displayed sample.

`SpriteMediaGetSpriteProperty` (page 171)
Retrieves the value of the specified sprite or sprite track property.

`SpriteMediaHitTestAllSprites` (page 172)
Determines whether any sprites are at a specified location.

`SpriteMediaHitTestOneSprite` (page 173)
Performs a hit testing operation on the sprite specified by a spriteID.

`SpriteMediaSetSpriteProperty` (page 179)
Sets the specified property of a sprite or sprite track.

`SpriteMediaSpriteIDToIndex` (page 180)
Converts a sprite ID to the corresponding sprite index.

`SpriteMediaSpriteIndexToID` (page 180)
Returns the ID of a sprite specified by a sprite index.

Managing Sprite Images Outside a Movie

`SpriteMediaDisposeImage` (page 163)
Frees the memory allocated for a sprite image outside a movie and removes that image from the sprite track in which it appears.

`SpriteMediaImageIDToIndex` (page 174)
Returns the index of an outside sprite image from the ID of that image.

`SpriteMediaImageIndexToID` (page 175)
Returns the ID of an outside sprite image from the index of that image.

`SpriteMediaNewImage` (page 175)
Creates a new movie sprite image outside a movie.

Managing the Video Frame Playback Rate

`VideoMediaGetStatistics` (page 198)
Returns the play-back frame rate of a movie.

`VideoMediaResetStatistics` (page 199)
Resets the video media handler's counters before using VideoMediaGetStatistics to determine the frame rate of a movie.

Movie Functions

`DisposeMovie` (page 36)
Frees any memory being used by a movie, including the memory used by the movie's tracks and media structures.

`NewMovie` (page 108)
Creates a new movie in memory.

`PutMovieForDataRefIntoHandle` (page 114)
Puts a self-contained movie into a handle.

Movie Posters and Movie Previews

`GetMoviePosterTime` (page 64)
Returns the poster's time in a movie.

`GetMoviePreviewMode` (page 66)
Determines whether a movie is in preview mode.

`GetMoviePreviewTime` (page 67)
Returns the starting time and duration of the movie's preview.

`PlayMoviePreview` (page 111)
Plays a movie's preview.

`SetMoviePosterTime` (page 142)
Sets the poster time for the movie.

`SetMoviePreviewMode` (page 144)
Places a movie into and out of preview mode.

`SetMoviePreviewTime` (page 145)
Defines the starting time and duration of the movie's preview.

`ShowMoviePoster` (page 161)
Displays a movie's poster.

Movie Toolbox Clock Support Functions

`AddCallBackToTimeBase` (page 26)
Places a callback event into the list of scheduled callback events.

`ExecuteCallBack` (page 40)
Called by a clock component when it determines that it is time to execute a callback function.

`GetFirstCallBack` (page 52)
Returns the first callback event associated with a specified time base.

`GetNextCallBack` (page 75)
Returns the next callback event associated with a specified time base.

`RemoveCallBackFromTimeBase` (page 133)
Removes a callback event from the list of scheduled callback events.

Movies and Your Event Loop

`CreateMovieControl` (page 33)
Creates a movie control object to pass to the Mac OS Control Manager.

`InvalidateMovieRegion` (page 90)
Invalidates a small area of a movie.

`IsMovieDone` (page 91)
Determines if a particular movie has completely finished playing.

`MoviesTask` (page 106)
Services active movies.

`QTGetTimeUntilNextTask` (page 119)
Reports the duration until the next time QuickTime needs to run a task.

`QTInstallNextTaskNeededSoonerCallback` (page 125)
Installs a QTNextTaskNeededSoonerCallbackProc callback.

`QTUninstallNextTaskNeededSoonerCallback` (page 132)
Removes a QTNextTaskNeededSoonerCallbackProc callback.

`UpdateMovie` (page 196)
Ensures that the Movie Toolbox properly displays your movie after it has been uncovered.

Preferred Movie Settings

`GetMoviePreferredRate` (page 65)
Returns a movie's default playback rate.

`GetMoviePreferredVolume` (page 66)
Returns a movie's preferred volume setting.

`SetMoviePreferredRate` (page 142)
Specifies a movie's default playback rate.

`SetMoviePreferredVolume` (page 143)
Sets a movie's preferred volume setting.

Saving Movies

`PutMovieIntoDataFork` (page 115)
Stores a movie in the data fork of a given file.

`PutMovieIntoHandle` (page 117)
Creates a new movie resource.

PutMovieIntoStorage (page 117)
Writes a movie to a storage location managed by a data handler.

Text Media Handler Functions

TextMediaAddHiliteSample (page 183)
Provides dynamic highlighting of text.

TextMediaAddTESample (page 184)
Specifies a TextEdit handle to be added to a specified media.

TextMediaAddTextSample (page 186)
Adds a single block of styled text to an existing media.

TextMediaFindNextText (page 189)
Searches for text with a specified media handler starting at a given time.

TextMediaHiliteTextSample (page 191)
Specifies selected text to be highlighted for a given text media handler.

TextMediaSetTextProc (page 194)
Specifies a custom function to be called whenever a text sample is displayed in a movie.

TextMediaSetTextSampleData (page 195)
Sets values before calling TextMediaAddTextSample or TextMediaAddTESample.

The Sound Description Structure

QTSoundDescriptionConvert (page 127)
Converts a sound description from one version to another.

Time Base Callback Functions

CallMeWhen (page 27)
Schedules a callback event.

CancelCallBack (page 29)
Cancels a callback event before it executes.

DisposeCallBack (page 35)
Disposes of a callback event.

GetCallBackTimeBase (page 51)
Retrieves the time base of a callback event.

GetCallBackType (page 52)
Retrieves a callback event's type.

NewCallBack (page 107)
Creates a new callback event.

Using the OpenGL Texture Context

`GetMovieVisualContext` (page 74)
Returns the current visual context for a movie.

`SetMovieVisualContext` (page 151)
Targets a movie to render into a visual context.

Working With Movie Spatial Characteristics

`DisposeMatte` (page 35)
Disposes of a matte obtained from the GetTrackMatte function.

`GetMovieBoundsRgn` (page 54)
Determines a movie's boundary region.

`GetMovieBox` (page 55)
Returns a movie's boundary rectangle, which is a rectangle that encompasses all of the movie's enabled tracks.

`GetMovieClipRgn` (page 57)
Determines a movie's clipping region.

`GetMovieDisplayBoundsRgn` (page 58)
Determines a movie's display boundary region.

`GetMovieDisplayClipRgn` (page 58)
Determines a movie's current display clipping region.

`GetMovieGWorld` (page 60)
Returns a movie's graphics world.

`GetMovieMatrix` (page 61)
Retrieves a movie's transformation matrix.

`GetTrackBoundsRgn` (page 85)
Lets the media limit the size of a track boundary rectangle.

`GetTrackClipRgn` (page 86)
Determines the clipping region of a track.

`GetTrackDisplayBoundsRgn` (page 86)
Determines the region a track occupies in a movie's graphics world.

`GetTrackMatte` (page 87)
Retrieves a copy of a track's matte.

`GetTrackMovieBoundsRgn` (page 88)
Determines the region the track occupies in a movie's boundary region.

`SetMovieBox` (page 135)
Sets a movie's boundary rectangle.

`SetMovieClipRgn` (page 136)
Establishes a movie's clipping region.

`SetMovieDisplayClipRgn` (page 137)
Establishes a movie's current display clipping region.

`SetMovieGWorld` (page 139)
Establishes a movie's display coordinate system by setting the graphics world for displaying the movie.

SetMovieMatrix (page 141)
Sets a movie's transformation matrix.

SetTrackClipRgn (page 159)
Sets the clipping region of a track.

SetTrackGWorld (page 160)
Forces a track to draw into a particular graphics world, which may be different from that of the movie.

SetTrackMatte (page 161)
Sets a track's matte.

Working With Movie Time

GetMovieDuration (page 59)
Returns the duration of a movie.

GetMovieRate (page 68)
Returns a movie's playback rate.

GetMovieTime (page 72)
Returns a movie's current time both as a time value and in a time structure.

GetMovieTimeBase (page 72)
Returns a movie's time base.

GetMovieTimeScale (page 73)
Returns the time scale of a movie.

SetMovieRate (page 145)
Sets a movie's playback rate.

SetMovieTime (page 148)
Changes a movie's current time.

SetMovieTimeScale (page 149)
Establishes a movie's time scale.

SetMovieTimeValue (page 149)
Sets a movie's time value.

Working With Progress and Cover Functions

SetMovieDrawingCompleteProc (page 137)
Assigns a drawing-complete function to a movie.

Working With Sound Descriptions

QTSoundDescriptionCreate (page 128)
Creates a sound description structure of the requested kind from an AudioStreamBasicDescription, optional audio channel layout, and optional magic cookie.

Working With Sound Volume

`GetMovieVolume` (page 75)
Returns a movie's current volume setting.

`SetMovieVolume` (page 151)
Sets a movie's current volume but does not store the setting in the movie.

Working With The Idle Manager

`QTIdleManagerClose` (page 120)
Closes the Mac OS Idle Manager.

`QTIdleManagerGetNextIdleTime` (page 121)
Retrieves the next idle time known to the Idle Manager.

`QTIdleManagerNeedsAnIdle` (page 121)
Tells the Idle Manager whether an idle will be required.

`QTIdleManagerOpen` (page 122)
Opens the Mac OS Idle Manager.

`QTIdleManagerSetNextIdleTime` (page 122)
Informs the idle manager of the next required idle time.

`QTIdleManagerSetNextIdleTimeDelta` (page 123)
Informs the idle manager of the time from the currently set idle time to the next idle time required after it.

`QTIdleManagerSetNextIdleTimeNever` (page 124)
Sets the next idle time indefinitely in the future.

`QTIdleManagerSetNextIdleTimeNow` (page 124)
Requests an idle as soon as possible.

`QTIdleManagerSetParent` (page 125)
Sets the parent of an Idle Manager instance.

Working With Time Base Values

`GetTimeBaseEffectiveRate` (page 77)
Returns the effective rate at which the specified time base is moving relative to its master clock.

`GetTimeBaseFlags` (page 77)
Obtains the control flags of a time base.

`GetTimeBaseRate` (page 80)
Retrieves the rate of a time base.

`GetTimeBaseStartTime` (page 82)
Determines the start time of a time base.

`GetTimeBaseStatus` (page 82)
Determines when the current time of a time base would fall outside of the range of values specified by the time base's start and stop times.

`GetTimeBaseStopTime` (page 83)
Determines the stop time of a time base.

GetTimeBaseTime (page 84)
Obtains the current time value from a time base.

SetTimeBaseFlags (page 152)
Sets the contents of the control flags of a time base.

SetTimeBaseRate (page 155)
Sets the rate of a time base.

SetTimeBaseStartTime (page 156)
Sets the start time of a time base.

SetTimeBaseStopTime (page 156)
Sets the stop time of a time base.

SetTimeBaseTime (page 157)
Sets the current time of a time base.

SetTimeBaseValue (page 158)
Sets the current time of a time base.

Working With Times

AddTime (page 26)
Adds two times.

ConvertTime (page 31)
Converts a time obtained from one time base into a time that is relative to another time base.

ConvertTimeScale (page 31)
Converts a time from one time scale into a time that is relative to another time scale.

SubtractTime (page 182)
Subtracts one time from another.

Working With User Data

GetMovieUserData (page 74)
Obtains access to a movie's user data list.

Working With Wired Sprites

SpriteMediaGetActionVariable (page 164)
Returns the value of the sprite track variable with the specified ID.

SpriteMediaSetActionVariable (page 177)
Sets the value of a sprite track variable to a specified value.

Supporting Functions

AttachTimeBaseToCurrentThread (page 27)
Attaches a time base to the current thread.

CheckQuickTimeRegistration (page 29)
Deprecated.

ConvertTimeToClockTime (page 32)
Converts a time record in a time base to clock time.

DetachTimeBaseFromCurrentThread (page 34)
Detaches a time base from the current thread.

EnterMoviesOnThread (page 39)
Indicates that the client will be using QuickTime on the current thread.

ExitMoviesOnThread (page 42)
Indicates to QuickTime that the client will no longer be using QuickTime on the current thread.

FlashMediaDoButtonActions (page 43)
Performs actions attached to a specified button.

FlashMediaFrameLabelToMovieTime (page 43)
Undocumented

FlashMediaFrameNumberToMovieTime (page 44)
Undocumented

FlashMediaGetDisplayedFrameNumber (page 45)
Undocumented

FlashMediaGetFlashVariable (page 45)
Gets the value of a specified Flash action variable.

FlashMediaGetRefConBounds (page 46)
Undocumented

FlashMediaGetRefConID (page 47)
Undocumented

FlashMediaGetSupportedSwfVersion (page 47)
Identifies the version of Flash that this version of QuickTime supports.

FlashMediaIDToRefCon (page 48)
Undocumented

FlashMediaSetFlashVariable (page 48)
Sets the specified Flash action variable to a value.

FlashMediaSetPan (page 49)
Undocumented

FlashMediaSetZoom (page 50)
Undocumented

FlashMediaSetZoomRect (page 50)
Undocumented

GetMovieAudioContext (page 54)
Returns the current audio context for a movie.

GetMovieNaturalBoundsRect (page 62)
Gets a movie's natural boundary rectangle.

GetMovieRateChangeConstraints (page 68)
Returns the minimum and maximum delay you can get when a movie's rate is changed.

GetNextTrackForCompositing (page 76)
Determines the next track in a movie's compositing process.

MovieMediaGetCurrentMovieProperty (page 103)
Retrieves current properties from a media handler's movie.

MovieMediaGetCurrentTrackProperty (page 104)
Retrieves the media type property from a media handler's track.

MovieMediaGetDoMCActionCallback (page 104)
Gets a DoMCActionProc callback for a media.

MovieMediaLoadChildMovieFromDataReference (page 105)
Undocumented

MovieMediaSetChildMovieDataReference (page 106)
Undocumented

NewMovieFromProperties (page 109)
Creates a new movie using movie properties.

PutMovieIntoDataFork64 (page 116)
Provides a 64-bit version of PutMovieIntoDataFork.

QTAudioContextCreateForAudioDevice (page 118)
Creates a QTAudioContext object that encapsulates a connection to a CoreAudio output device.

QTParseTextHREF (page 126)
Undocumented

QTSoundDescriptionGetProperty (page 129)
Gets a particular property of a sound description.

QTSoundDescriptionGetPropertyInfo (page 130)
Gets information about a particular property of a sound description.

QTSoundDescriptionSetProperty (page 130)
Sets a particular property of a sound description.

QTTextToNativeText (page 131)
Undocumented

SetMovieAudioContext (page 135)
Targets a movie to render into an audio context.

SetMovieVideoOutput (page 150)
Indicates to the ICM the video output component being used with a given movie.

SetTimeBaseOffsetTimeBase (page 155)
Attaches an offset time base to another time base.

SpriteMediaDisposeSprite (page 164)
Disposes of memory allocated for a sprite.

SpriteMediaGetActionVariableAsString (page 165)
Undocumented

SpriteMediaGetProperty (page 168)
Gets a sprite property; superseded by SpriteMediaGetSpriteProperty.

SpriteMediaGetSpriteActionsForQTEvent (page 169)
Gets the sprite action atom for an event.

SpriteMediaHitTestSprites (page 173)
Undocumented

SpriteMediaNewSprite (page 176)
Creates a new sprite.

`SpriteMediaSetActionVariableToString` (page 177)
Undocumented

`SpriteMediaSetProperty` (page 178)
Sets a sprite property; superseded by SpriteMediaSetSpriteProperty.

`TextMediaDrawRaw` (page 188)
Undocumented

`TextMediaGetTextProperty` (page 190)
Sets properties of a text media.

`TextMediaRawIdle` (page 192)
Undocumented

`TextMediaRawSetup` (page 193)
Undocumented

`TextMediaSetTextProperty` (page 194)
Sets properties of a text media.

`VideoMediaGetCodecParameter` (page 197)
Undocumented

`VideoMediaGetStallCount` (page 198)
Undocumented

`VideoMediaSetCodecParameter` (page 200)
Undocumented

Functions

AbortPrePrerollMovie

Terminates the operation of PrePrerollMovie.

```
void AbortPrePrerollMovie (
   Movie m,
   OSErr err
);
```

Parameters

m
The movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie` (page 108), `NewMovieFromFile`, and `NewMovieFromHandle`.

err
See `Error Codes`. Returns `noErr` if there is no error.

Discussion

You normally call this function only if you have previously called `PrePrerollMovie` (page 112) asynchronously and the user quits your application.

Version Notes

Introduced in QuickTime 4.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Movies.h`

AddCallBackToTimeBase

Places a callback event into the list of scheduled callback events.

```
OSErr AddCallBackToTimeBase (
   QTCallBack cb
);
```

Parameters

cb

Specifies the callback event for the operation. Your clock component obtains this value from the parameters passed to `ClockCallMeWhen`.

Return Value
You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Discussion
If your component calls this function, the Movie Toolbox notifies it of time, rate, or stop and start changes via `ClockRateChanged` and `ClockTimeChanged`.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Movies.h`

AddTime

Adds two times.

```
void AddTime (
   TimeRecord *dst,
   const TimeRecord *src
);
```

Parameters

dst

A pointer to a time structure. This time structure contains one of the operands for the addition. `AddTime` returns the result of the addition into this time structure.

src

A pointer to a time structure. The Movie Toolbox adds this value to the time or duration specified by the `dst` parameter.

Return Value
You can access this function's error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71).

Discussion
You must specify the input times in time structures. The result value is formatted as a duration or a time value, the same as the format of the structure pointed to by the `dst` parameter.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Movies.h`

AttachTimeBaseToCurrentThread

Attaches a time base to the current thread.

```
OSErr AttachTimeBaseToCurrentThread (
   TimeBase tb
);
```

Parameters

tb
A time base.

Return Value
See `Error Codes` in the QuickTime API Reference. Returns `noErr` if there is no error.

Version Notes
Introduced in QuickTime 6.4.

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

Declared In
`Movies.h`

CallMeWhen

Schedules a callback event.

```
OSErr CallMeWhen (
   QTCallBack cb,
   QTCallBackUPP callBackProc,
   long refCon,
   long param1,
   long param2,
   long param3
);
```

Parameters

cb

The callback event for the operation. You obtain this identifier from `NewCallBack` (page 107).

callBackProc

Points to your callback function, described in `QTCallBackProc`.

refCon

A reference constant to be passed to your callback. Use this parameter to point to a data structure containing any information your function needs.

param1

Contains scheduling information. The Movie Toolbox interprets this parameter based on the value of the `cbType` parameter to `NewCallBack` (page 107). If `cbType` is set to `callBackAtTime`, the param1 parameter contains flags (see below) indicating when to invoke your callback function for this callback event. If the `cbType` parameter is set to `callBackAtRate`, param1 contains flags (see below) indicating when to invoke your callback function for this event. Be sure to set unused flags to 0.

param2

Contains scheduling information. The Movie Toolbox interprets this parameter based on the value of the `cbType` parameter to `NewCallBack` (page 107). If `cbType` is set to `callBackAtTime`, the param2 parameter contains the time value at which your callback function is to be invoked for this event. The param1 parameter contains flags affecting when the Movie Toolbox calls your function. If `cbType` is set to `callBackAtRate`, the param2 parameter contains the rate value at which your callback function is to be invoked for this event. The param1 parameter contains flags affecting when the Movie Toolbox calls your function.

param3

The time scale in which to interpret the time value that is stored in param3 if `cbType` is set to `callBackAtTime`.

Return Value

You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Discussion

You can call this function from your callback function.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

qtbigscreen

qtbigscreen.win

Show Movie

SimpleCocoaMovie

SimpleCocoaMovieQT

Declared In
Movies.h

CancelCallBack

Cancels a callback event before it executes.

```
void CancelCallBack (
   QTCallBack cb
);
```

Parameters

cb

The callback event for this operation. You obtain this value from NewCallBack (page 107).

Return Value
You can access this function's error returns through GetMoviesError (page 70) and GetMoviesStickyError (page 71).

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
Movies.h

CheckQuickTimeRegistration

Deprecated.

```
void CheckQuickTimeRegistration (
   void *registrationKey,
   long flags
);
```

Version Notes
This function is listed for historical purposes only. It may be unsupported or removed in future versions of QuickTime.

Availability
Available in Mac OS X v10.0 and later.

Declared In
Movies.h

ChooseMovieClock

Searches media handlers to find the best clock for a movie.

```
void ChooseMovieClock (
   Movie m,
   long flags
);
```

Parameters

m

The movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie` (page 108), `NewMovieFromFile`, and `NewMovieFromHandle`.

flags

Currently not used; set to 0.

Discussion

This function calls `MediaGetClock` and finds the first media handler that has a custom clock. It then calls `SetMovieMasterClock` (page 139) to use the best clock as the movie's master timebase clock.

`ChooseMovieClock` can be used to tie the movie's master timebase to a sound clock if there is a sound track. If there is no sound track, the microseconds clock is used as the master timebase.

Version Notes

Introduced in QuickTime 6.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`Movies.h`

ClearMoviesStickyError

Clears the sticky error value.

```
void ClearMoviesStickyError (
   void
);
```

Discussion

The Movie Toolbox provides two error values to your application: the current error and the sticky error. The current error is the result code from the last Movie Toolbox function; it is updated each time your application calls a Movie Toolbox function. The Movie Toolbox saves the same result code in the sticky error value. Your application clears the sticky error value by calling `ClearMoviesStickyError`. The Movie Toolbox then places the first nonzero result code from any toolbox function used by your application into the sticky error value. The Movie Toolbox does not update the sticky error value until your application clears it again.

Special Considerations

Many Movie Toolbox functions don't return an error as a function result; you must use `GetMoviesError` to obtain the result code. Even if a function explicitly returns an error as a function result, that result is also available using `GetMoviesError`. The Movie Toolbox does not place a result code into the sticky error value until the field has been cleared. Your application is responsible for clearing the sticky error value to ensure that it does not contain a stale result code.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
BurntTextSampleCode
vrmakeobject
vrmakepano
VRMakePano Library
vrmakepano.win

Declared In
`Movies.h`

ConvertTime

Converts a time obtained from one time base into a time that is relative to another time base.

```
void ConvertTime (
   TimeRecord *theTime,
   TimeBase newBase
);
```

Parameters

`theTime`

A pointer to a time structure that contains the time value to be converted. The `ConvertTime` function replaces the contents of this time structure with the time value relative to the specified time base.

`newBase`

The time base for this operation. Your application obtains this time base identifier from the `NewTimeBase` (page 110) function.

Return Value
You can access this function's error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71).

Discussion
This function includes the rate associated with each time value in the conversion; therefore, you should use this function when you want to convert time values. Both time bases must rely on the same time source, and you must specify the time to be converted in a time structure.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Movies.h`

ConvertTimeScale

Converts a time from one time scale into a time that is relative to another time scale.

```
void ConvertTimeScale (
   TimeRecord *theTime,
   TimeScale newScale
);
```

Parameters

theTime

A pointer to a time structure that contains the time value to be converted. `ConvertTimeScale` replaces the contents of this time structure with the time value relative to the specified time scale.

newScale

The time scale for this operation.

Return Value

You can access this function's error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71).

Discussion

This function does not include the rate associated with the time value in the conversion; therefore, you should use this function when you want to convert time durations, but not when converting time values.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

AlwaysPreview

qtsndtween

qtsndtween.win

qttext.win

TimeCode Media Handlers

Declared In

`Movies.h`

ConvertTimeToClockTime

Converts a time record in a time base to clock time.

```
void ConvertTimeToClockTime (
   TimeRecord *time
);
```

Parameters

time

The `TimeRecord` structure to be converted. It must contain a valid time base; otherwise it remains untouched.

Discussion

The result of this call has no meaning it the time base rate is 0.

Version Notes

Introduced in QuickTime 6.4.

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

Declared In
`Movies.h`

CreateMovieControl

Creates a movie control object to pass to the Mac OS Control Manager.

```
OSErr CreateMovieControl (
   WindowRef theWindow,
   Rect *localRect,
   Movie theMovie,
   UInt32 options,
   ControlRef *returnedControl
);
```

Parameters

theWindow

The window in which the control is placed.

localRect

A pointer to a `Rect` structure that describes in local coordinates the window in which the movie control is placed. If `NIL` is passed, the movie control is positioned at 0,0 within the window; it will have the natural dimensions of the movie plus the height of the movie controls if they are visible. If 0 height and width is passed, this parameter is interpreted as an anchor point and the top left point of the movie control will be located at this position with height and width as in the `NIL` case. For all other cases of rectangles, the movie control is centered within the rectangle by default and is sized to fit within it while maintaining the movie's aspect ratio.

theMovie

The movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie` (page 108), `NewMovieFromFile`, and `NewMovieFromHandle`.

options

Constants (see below) that determine parts of the movie control's appearance. See these constants:

`kMovieControlOptionHideController`
`kMovieControlOptionLocateTopLeft`
`kMovieControlOptionEnableEditing`
`kMovieControlOptionHandleEditingHI`
`kMovieControlOptionSetKeysEnabled`
`kMovieControlOptionManuallyIdled`

returnedControl

A handle to a `ControlRecord` struct. This defines a movie control, suitable for passing to Mac OS Control Manager functions.

Return Value
You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`. This routine returns an error if there is a problem with one of the parameters or if an error occurred while creating the underlying movie controller or the custom control itself. If an error is returned, the value of `returnedControl` is undefined.

Discussion
The Carbon Movie Control is implemented as a custom control, which installs an event handler to handle the Carbon Events sent to controls. When a Carbon Movie Control is created for a movie, a movie controller is also created. The movie control then directs user interface events to the controller. The application can install event handlers on the Carbon Movie Control to handle such things as contextual menu clicks or to intercept events to do special processing. Control Manager calls can be made as well.

Special Considerations

The control can be deleted by calling the Mac OS function `DisposeControl`. Note that the control is automatically disposed of if the enclosing window is destroyed. Note, too, that the underlying movie controller is disposed of when the control is deleted.

Version Notes
Introduced in QuickTime 6.

Availability
Available in Mac OS X v10.2 and later.

Related Sample Code
QTCarbonShell
qtshellCEvents
qtshellCEvents.win
QuickTimeMovieControl

Declared In
`Movies.h`

DetachTimeBaseFromCurrentThread

Detaches a time base from the current thread.

```
OSErr DetachTimeBaseFromCurrentThread (
   TimeBase tb
);
```

Parameters

tb
A time base.

Return Value
See `Error Codes` in the QuickTime API Reference. Returns `noErr` if there is no error.

Version Notes
Introduced in QuickTime 6.4.

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

Declared In
`Movies.h`

DisposeCallBack

Disposes of a callback event.

```
void DisposeCallBack (
   QTCallBack cb
);
```

Parameters

cb

The callback event for the operation. You obtain this value from `NewCallBack` (page 107).

Return Value

You can access this function's error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71).

Discussion

You should call this function when you are done with each callback event.

Special Considerations

Don't call this function at interrupt time.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

qtbigscreen

qtbigscreen.win

Show Movie

SimpleCocoaMovie

SimpleCocoaMovieQT

Declared In

`Movies.h`

DisposeMatte

Disposes of a matte obtained from the GetTrackMatte function.

```
void DisposeMatte (
   PixMapHandle theMatte
);
```

Parameters

theMatte

`Handle` to the matte to be disposed. Your application obtains this handle from `GetTrackMatte` (page 87).

Return Value

You can access this function's error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71).

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
Inside Mac Movie TB Code

Declared In
`Movies.h`

DisposeMovie

Frees any memory being used by a movie, including the memory used by the movie's tracks and media structures.

```
void DisposeMovie (
   Movie theMovie
);
```

Parameters

theMovie

Identifies the movie to be freed. Your application obtains this movie identifier from such functions as `NewMovie` (page 108), `NewMovieFromFile`, or `NewMovieFromHandle`.

Return Value
You can access this function's error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71).

Discussion
Your application should call this function when it is done working with a movie, as shown in the following example:

```
// DisposeMovie coding example
// See "Discovering QuickTime," page 85
void CreateMyCoolMovie (void)
{
    StandardFileReply   sfr;
    Movie               movie =NIL;
    FSSpec              fss;
    short               nFileRefNum =0;
    short               nResID =movieInDataForkResID;
    StandardPutFile("\pEnter movie file name:", "\puntitled.mov", &sfr);
    if (!sfr.sfGood)
        return;
    CreateMovieFile(&sfr.sfFile,
                    FOUR_CHAR_CODE('TVOD'),
                    smCurrentScript,
                    createMovieFileDeleteCurFile |
createMovieFileDontCreateResFile,
                    &nFileRefNum,
                    &movie);
    CreateMyVideoTrack(movie);      // See "Creating a Track," below
    AddMovieResource(movie, nFileRefNum, &nResID, NIL);
    if (nFileRefNum !=0)
```

```
        CloseMovieFile(nFileRefNum);
    DisposeMovie(movie);
}
```

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
MakeEffectMovie
vrmakeobject
vrmakepano
VRMakePano Library
vrmakepano.win

Declared In
`Movies.h`

DisposeTimeBase

Disposes of a time base once you are finished with it.

```
void DisposeTimeBase (
   TimeBase tb
);
```

Parameters

tb

The time base for this operation. Your application obtains this time base identifier from `NewTimeBase` (page 110).

Return Value
You can access this function's error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71).

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
qteffects
qteffects.win
qtshoweffect
qtshoweffect.win
vrscript.win

Declared In
`Movies.h`

EnterMovies

Initializes the Movie Toolbox and creates a private storage area for your application.

```
OSErr EnterMovies (
   void
);
```

Return Value
Be sure to check the value returned by this function before using any other facilities of the Movie Toolbox. See `Error Codes`. Returns `noErr` if there is no error.

Discussion
Before calling any Movie Toolbox functions, you must use `EnterMovies` to initialize the toolbox. Your application may call `EnterMovies` multiple times. The following code sample demonstrates how your application can call the Gestalt Manager to determine whether the Movie Toolbox is installed, using the selector `gestaltQuickTime` (`'qtim'`), before calling `EnterMovies`:

```
//Using the Gestalt Manager with the Movie Toolbox
#include <GestaltEqu.h>
#include <Movies.h>
Boolean IsQuickTimeInstalled (void)
{
    short   error;
    long    result;

    error =Gestalt (gestaltQuickTime, &result);
    return (error ==noErr);
}
void main (void)
{
    Boolean qtInstalled;
    .
    .
    .
    qtInstalled =IsQuickTimeInstalled ();
}
// EnterMovies coding example
// See "Discovering QuickTime," page 242
void MyInitMovieToolbox (void)
{
    InitGraf(&qd.thePort);
    InitFonts();
    InitWindows();
    InitMenus();
    TEInit();
    InitDialogs(NIL);
    EnterMovies();
}
void main (void)
{
    MyInitMovieToolbox();
    CreateMyCoolMovie();
}
```

Special Considerations

You should initialize any other Macintosh managers your application uses before calling `EnterMovies`. You do not need to balance calls to `EnterMovies` with calls to `ExitMovies` (page 40); you need to call `ExitMovies` only if you finish with the Movie Toolbox long before your application is ready to quit.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
Inside Mac Movie TB Code
MakeEffectMovie
mfc.win
MovieGWorlds
SGDataProcSample

Declared In
`Movies.h`

EnterMoviesOnThread

Indicates that the client will be using QuickTime on the current thread.

```
OSErr EnterMoviesOnThread (
   UInt32 inFlags
);
```

Parameters

inFlags

Flag (see below) indicating how the executing thread will use QuickTime. Setting the thread mode is a convenience provided by this function. Pass 0 for the default options. See these constants:
`kQTEnterMoviesFlagDontSetComponentsThreadMode`

Return Value
See `Error Codes` in the QuickTime API Reference. Returns `noErr` if there is no error. This function returns an appropriate operating system or QuickTime error if the operation couldn't be completed. This might occur because a second call on the thread was made that used incompatible flags (for example, the first call required a shared state but a subsequent call required a private state).

Discussion
This function is analogous to `EnterMovies`. It initializes QuickTime and prepares QuickTime for calls from its thread. Unlike `EnterMovies`, this function allows the client to indicate if its access to QuickTime requires sharing of QuickTime state with the main thread. The default is to maintain a private state.

Version Notes
Introduced in QuickTime 6.4.

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

Related Sample Code
ExtractMovieAudioToAIFF
QTAudioExtractionPanel
ThreadsExportMovie
ThreadsImporter
ThreadsImportMovie

Declared In
`Movies.h`

ExecuteCallBack

Called by a clock component when it determines that it is time to execute a callback function.

```
void ExecuteCallBack (
   QTCallBack cb
);
```

Parameters

cb

Specifies the callback event for the operation. Your clock component obtains this value from the parameters passed to your `ClockCallMeWhen` function.

Return Value
You can access this function's error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71).

Discussion
Before calling the application's function, the `ExecuteCallBack` function cancels the callback event. In this manner, the callback event is prevented from executing twice in succession. It is up to the application, or the callback function itself, to reschedule the callback event.

Special Considerations

Your clock component should not release the memory associated with the callback event at this time. You should do so only with `ClockDisposeCallBack`. This is particularly important when a callback function cannot execute at interrupt time, since the Movie Toolbox schedules such functions for invocation at a later time.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Movies.h`

ExitMovies

Automatically called when an application quits.

```
void ExitMovies (
   void
);
```

Return Value
You can access this function's error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71).

Discussion
You only need to call this function if you finish with the Movie Toolbox long before your application is ready to quit. When you call `ExitMovies`, the Movie Toolbox releases the private storage (which may be significant) that was allocated when you called `EnterMovies` (page 38). As a general rule, your application seldom uses this function; the following code illustrates an exception:

```
// ExitMovies coding example
// See "Discovering QuickTime," page 225
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    LPSTR lpCmdLine, int nCmdShow)
{
    MSG         msg;
    HANDLE      hAccelTable;

    if  (!hPrevInstance)                  // Is there a previous instance?
        if (!(InitApplication(hInstance)))      // Register window class
            return FALSE;                       // Report failure

    if (InitializeQTML(0) !=0) {                        // Initialize QTML
        MessageBox(hwnd, "QuickTime not available",     // Notify user
                         "", MB_OK);
        return FALSE;                                   // Report failure
    }  // end if (InitializeQTML(0) !=0)

    if (EnterMovies() !=0) {                        // Initialize QuickTime
        MessageBox(hwnd, "QuickTime not available",     // Notify user
                         "", MB_OK);
        return FALSE;                                   // Report failure
    }  // end if (EnterMovies() !=0)

    if (!(InitInstance(hInstance, nCmdShow)))         // Create main window
        return FALSE;                                 // Report failure

    hAccelTable =LoadAccelerators(hInstance,     // Load accelerator table
                 MAKEINTRESOURCE(IDR_ACCELSIMPLESDI));

    //  Main message loop

    while (GetMessage(&msg, NIL, 0, 0))        // Retrieve next message
        if (!TranslateAccelerator(msg.hwnd,    // Check for kbd accelerator
                        hAccelTable, &msg)) {
            TranslateMessage(&msg);     // Convert virtual key to character
            DispatchMessage(&msg);      // Send message to window procedure
        }  // end if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg))

    ExitMovies();                                   // Terminate Toolbox
    TerminateQTML();                                // Terminate QuickTime

    return msg.wParam;
}  // end WinMain
```

Special Considerations

Before calling `ExitMovies`, be sure that you have closed your connections to any components that use the Movie Toolbox, such as movie controllers, sequence grabbers, and so on.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
MakeEffectMovie
makeeffectslideshow
mfc.win
qtcontroller
qtwiredactions

Declared In
`Movies.h`

ExitMoviesOnThread

Indicates to QuickTime that the client will no longer be using QuickTime on the current thread.

```
OSErr ExitMoviesOnThread (
   void
);
```

Return Value
See `Error Codes` in the QuickTime API Reference. Returns `noErr` if there is no error. Returns an appropriate operating system or QuickTime error if the operation couldn't be completed. This might occur because a previous call to `EnterMoviesOnThread` was not made.

Discussion
This function should be called before exiting from a spawned thread that uses QuickTime. It undoes the setup performed by `EnterMoviesOnThread`. Each call to `EnterMoviesOnThread` should be matched with a call to this function. This function should not be called on a thread without a previous call to `EnterMoviesOnThread`.

Version Notes
Introduced in QuickTime 6.4.

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

Related Sample Code
ExtractMovieAudioToAIFF
QTAudioExtractionPanel
ThreadsExportMovie
ThreadsImporter
ThreadsImportMovie

Declared In
Movies.h

FlashMediaDoButtonActions

Performs actions attached to a specified button.

```
ComponentResult FlashMediaDoButtonActions (
   MediaHandler mh,
   char *path,
   long buttonID,
   long transition
);
```

Parameters

mh
The Toolbox's connection to your derived Flash media handler. You can obtain this reference from GetMediaHandler.

path
Specifies the path to the button to which the action is attached.

buttonID
The ID of the button.

transition
Sends a mouse transition message to the object and whatever Flash actions are associated with that transition on the object that should be performed. The values are specific Flash transition constants.

Return Value
You can access Movie Toolbox error returns through GetMoviesError (page 70) and GetMoviesStickyError (page 71), as well as in the function result. See Error Codes.

Version Notes
Introduced in QuickTime 5.

Availability
Available in Mac OS X v10.0 and later.

Declared In
Movies.h

FlashMediaFrameLabelToMovieTime

Undocumented

```
ComponentResult FlashMediaFrameLabelToMovieTime (
   MediaHandler mh,
   Ptr theLabel,
   TimeValue *movieTime
);
```

Parameters

mh

The Toolbox's connection to your derived Flash media handler. You can obtain this reference from `GetMediaHandler`.

theLabel

Undocumented

movieTime

Undocumented

Return Value

You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Version Notes

Introduced in QuickTime 4.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Movies.h`

FlashMediaFrameNumberToMovieTime

Undocumented

```
ComponentResult FlashMediaFrameNumberToMovieTime (
   MediaHandler mh,
   long flashFrameNumber,
   TimeValue *movieTime
);
```

Parameters

mh

The Toolbox's connection to your derived Flash media handler. You can obtain this reference from `GetMediaHandler`.

flashFrameNumber

Undocumented

movieTime

Undocumented

Return Value

You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Version Notes

Introduced in QuickTime 4.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Movies.h`

FlashMediaGetDisplayedFrameNumber

Undocumented

```
ComponentResult FlashMediaGetDisplayedFrameNumber (
   MediaHandler mh,
   long *flashFrameNumber
);
```

Parameters

mh

The Toolbox's connection to your derived Flash media handler. You can obtain this reference from `GetMediaHandler`.

flashFrameNumber

Undocumented

Return Value
You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Version Notes
Introduced in QuickTime 4.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Movies.h`

FlashMediaGetFlashVariable

Gets the value of a specified Flash action variable.

```
ComponentResult FlashMediaGetFlashVariable (
   MediaHandler mh,
   char *path,
   char *name,
   Handle *theVariableCStringOut
);
```

Parameters

mh

The Toolbox's connection to your derived Flash media handler. You can obtain this reference from `GetMediaHandler`.

path

Specifies the path to the Flash button to which the variable is attached.

name

Specifies the name of the Flash variable.

theVariableCStringOut

A handle to the value of the Flash variable as a C string.

Return Value

You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Version Notes

Introduced in QuickTime 5.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Movies.h`

FlashMediaGetRefConBounds

Undocumented

```
ComponentResult FlashMediaGetRefConBounds (
   MediaHandler mh,
   long refCon,
   long *left,
   long *top,
   long *right,
   long *bottom
);
```

Parameters

mh

The Toolbox's connection to your derived Flash media handler. You can obtain this reference from `GetMediaHandler`.

refCon

Undocumented

left

Undocumented

top

Undocumented

right

Undocumented

bottom

Undocumented

Return Value

You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Version Notes

Introduced in QuickTime 4.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Movies.h`

FlashMediaGetRefConID

Undocumented

```
ComponentResult FlashMediaGetRefConID (
   MediaHandler mh,
   long refCon,
   long *refConID
);
```

Parameters

mh

The Toolbox's connection to your derived Flash media handler. You can obtain this reference from `GetMediaHandler`.

refCon

Undocumented

refConID

Undocumented

Return Value
You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Version Notes
Introduced in QuickTime 4.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Movies.h`

FlashMediaGetSupportedSwfVersion

Identifies the version of Flash that this version of QuickTime supports.

```
ComponentResult FlashMediaGetSupportedSwfVersion (
   MediaHandler mh,
   unsigned char *swfVersion
);
```

Parameters

mh

The Toolbox's connection to your derived Flash media handler. You can obtain this reference from `GetMediaHandler`.

swfVersion

The version number of the most current version of Flash that this version of QuickTime supports.

Return Value
You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Version Notes
Introduced in QuickTime 5.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Movies.h`

FlashMediaIDToRefCon

Undocumented

```
ComponentResult FlashMediaIDToRefCon (
   MediaHandler mh,
   long refConID,
   long *refCon
);
```

Parameters

mh

The Toolbox's connection to your derived Flash media handler. You can obtain this reference from `GetMediaHandler`.

refConID

Undocumented

refCon

Undocumented

Return Value
You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Version Notes
Introduced in QuickTime 4.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Movies.h`

FlashMediaSetFlashVariable

Sets the specified Flash action variable to a value.

```
ComponentResult FlashMediaSetFlashVariable (
   MediaHandler mh,
   char *path,
   char *name,
   char *value,
   Boolean updateFocus
);
```

Parameters

mh

The Toolbox's connection to your derived Flash media handler. You can obtain this reference from `GetMediaHandler`.

path

Specifies the path to the Flash button to which the variable is attached.

name

Specifies the name of the Flash variable.

value

Specifies the new value of the Flash variable.

updateFocus

Pass TRUE if the focus is to be changed.

Return Value

You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Version Notes

Introduced in QuickTime 5.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Movies.h`

FlashMediaSetPan

Undocumented

```
ComponentResult FlashMediaSetPan (
   MediaHandler mh,
   short xPercent,
   short yPercent
);
```

Parameters

mh

Undocumented

xPercent

Undocumented

yPercent

Undocumented

Return Value
You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Version Notes
Introduced in QuickTime 4.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Movies.h`

FlashMediaSetZoom

Undocumented

```
ComponentResult FlashMediaSetZoom (
   MediaHandler mh,
   short factor
);
```

Parameters

mh
Undocumented

factor
Undocumented

Return Value
You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Version Notes
Introduced in QuickTime 4.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Movies.h`

FlashMediaSetZoomRect

Undocumented

```
ComponentResult FlashMediaSetZoomRect (
   MediaHandler mh,
   long left,
   long top,
   long right,
   long bottom
);
```

Parameters

mh

Undocumented

left

Undocumented

top

Undocumented

right

Undocumented

bottom

Undocumented

Return Value

You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Version Notes

Introduced in QuickTime 4.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Movies.h`

GetCallBackTimeBase

Retrieves the time base of a callback event.

```
TimeBase GetCallBackTimeBase (
   QTCallBack cb
);
```

Parameters

cb

The callback event for the operation. You obtain this value from the `NewCallBack` (page 107) function.

Return Value

A pointer to a `TimeBaseRecord` structure.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In
`Movies.h`

GetCallBackType

Retrieves a callback event's type.

```
short GetCallBackType (
   QTCallBack cb
);
```

Parameters

cb

The callback event for the operation. You obtain this value from `NewCallBack` (page 107).

Return Value
The callback type constant (see below). If the high-order bit (defined by `callBackAtInterrupt`) of the returned value is set to 1, the event can be invoked at interrupt time.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Movies.h`

GetFirstCallBack

Returns the first callback event associated with a specified time base.

```
QTCallBack GetFirstCallBack (
   TimeBase tb
);
```

Parameters

tb

Specifies the time base for the operation. Your component can obtain the time base reference from your `ClockSetTimeBase` function or from the Movie Toolbox's `GetCallBackTimeBase` (page 51) function.

Return Value
A pointer to a `CallBackRecord` structure. Your software can pass this structure to other functions, such as `ClockRateChanged`.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Movies.h`

GetMovieActive

Determines whether a movie is currently active.

```
Boolean GetMovieActive (
   Movie theMovie
);
```

Parameters

theMovie

The movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie` (page 108), `NewMovieFromFile`, and `NewMovieFromHandle`.

Return Value

TRUE if the movie is currently active, FALSE otherwise.

Special Considerations

The Movie Toolbox services only active movies.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Movies.h`

GetMovieActiveSegment

Determines what portion of a movie is currently active for playing.

```
void GetMovieActiveSegment (
   Movie theMovie,
   TimeValue *startTime,
   TimeValue *duration
);
```

Parameters

theMovie

The movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie` (page 108), `NewMovieFromFile`, and `NewMovieFromHandle`.

startTime

A pointer to a time value. `GetMovieActiveSegment` places the starting time of the active segment into the field referred to by this parameter. If the returned time value is set to -1, the entire movie is active. In this case, the Movie Toolbox does not return any duration information.

duration

A pointer to a time value. `GetMovieActiveSegment` places the duration of the active movie segment into the field referred to by this parameter. If the entire movie is active, the `startTime` parameter is set to -1 and this parameter does not return any duration information.

Return Value

You can access this function's error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71).

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Movies.h`

GetMovieAudioContext

Returns the current audio context for a movie.

```
OSStatus GetMovieAudioContext (
   Movie movie,
   QTAudioContextRef *audioContext
);
```

Parameters

movie
The movie.

audioContext
A pointer to a variable to receive the audio context.

Return Value
An error code. Returns `noErr` if there is no error.

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

Declared In
`Movies.h`

GetMovieBoundsRgn

Determines a movie's boundary region.

```
RgnHandle GetMovieBoundsRgn (
   Movie theMovie
);
```

Parameters

theMovie
The movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie` (page 108), `NewMovieFromFile`, and `NewMovieFromHandle`.

Return Value
A handle to a `MacRegion` structure that the function allocates. If the movie does not have a spatial representation at the current time, the function returns an empty region. If the function could not satisfy the request, it sets the returned handle to `NIL`.

Discussion
The Movie Toolbox derives the boundary region only from enabled tracks, and only from those tracks that are used in the current display mode (that is, movie or preview). The boundary region is valid for the current movie time.

Special Considerations

Your application must dispose of the returned region when it is done with it.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Movies.h`

GetMovieBox

Returns a movie's boundary rectangle, which is a rectangle that encompasses all of the movie's enabled tracks.

```
void GetMovieBox (
   Movie theMovie,
   Rect *boxRect
);
```

Parameters

theMovie

The movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie` (page 108), `NewMovieFromFile`, and `NewMovieFromHandle`.

boxRect

A pointer to a rectangle. `GetMovieBox` returns the coordinates of the movie's boundary rectangle into the structure referred to by this parameter.

Return Value
You can access this function's error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71).

Discussion
The movie box is in the coordinate system of the movie's graphics world and defines the movie's boundaries over the entire duration of the movie. The movie's boundary rectangle defines the size and shape of the movie before the Movie Toolbox applies the display clipping region. The following code sample illustrates the use of `GetMovieBox`:

```
// GetMovieBox coding example
// See "Discovering QuickTime," page 218
void main (void)
{
    WindowRef       pMacWnd;
    Rect            rectWnd;
    Rect            rectMovie;
    Movie           movie;
    Boolean         bDone =FALSE;
    OSErr           nErr;
```

```
    EventRecord     er;
    WindowRef       pWhichWnd;
    short           nPart;
    InitGraf(&qd.thePort);
    InitFonts();
    InitWindows();
    InitMenus();
    TEInit();
    InitDialogs(NIL);
    nErr =EnterMovies();
    if (nErr !=noErr)
        return;

    SetRect(&rectWnd, 100, 100, 200, 200);
    pMacWnd =NewCWindow(NIL, &rectWnd, "\pMovie", FALSE,
                        noGrowDocProc, (WindowRef)-1, TRUE, 0);
    SetPort(pMacWnd);
    movie =GetMovie();
    if (movie ==NIL)
        return;

    GetMovieBox(movie, &rectMovie);
    OffsetRect(&rectMovie, -rectMovie.left, -rectMovie.top);
    SetMovieBox(movie, &rectMovie);

    SizeWindow(pMacWnd, rectMovie.right, rectMovie.bottom, TRUE);
    ShowWindow(pMacWnd);
    SetMovieGWorld(movie, (CGrafPtr)pMacWnd, NIL);

    StartMovie(movie);

    . . .

    DisposeMovie(movie);
    DisposeWindow(pMacWnd);
}
```

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
MakeEffectMovie
qteffects
vrmovies
vrscript
vrscript.win

Declared In
Movies.h

GetMovieClipRgn

Determines a movie's clipping region.

```
RgnHandle GetMovieClipRgn (
   Movie theMovie
);
```

Parameters

theMovie

The movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie` (page 108), `NewMovieFromFile`, and `NewMovieFromHandle`.

Return Value

A handle to a `MacRegion` structure, which the function allocates, that represents the clipping region. If the function could not satisfy your request or if there is no clipping region defined for the movie, `GetMovieClipRgn` sets the returned handle to `NIL`.

Discussion

The clipping region is saved with the movie when your application saves the movie.

Special Considerations

Your application must dispose of this region when it is done with it.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Movies.h`

GetMovieCreationTime

Returns the movie's creation date and time information.

```
unsigned long GetMovieCreationTime (
   Movie theMovie
);
```

Parameters

theMovie

The movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie` (page 108), `NewMovieFromFile`, and `NewMovieFromHandle`.

Return Value

The movie's creation date and time.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In
Movies.h

GetMovieDisplayBoundsRgn

Determines a movie's display boundary region.

```
RgnHandle GetMovieDisplayBoundsRgn (
   Movie theMovie
);
```

Parameters

theMovie

The movie for this operation. Your application obtains this movie identifier from such functions as NewMovie (page 108), NewMovieFromFile, and NewMovieFromHandle.

Return Value
A handle to a MacRegion structure that the function allocates. If the movie does not have a spatial representation at the current time, the function returns an empty region. If the function could not satisfy the request, it sets the returned handle to NIL.

Discussion
The display boundary region encloses all of a movie's enabled tracks after the track matrix, track clip, movie matrix, and movie clip have been applied to them. This region is in the display coordinate system of the movie's graphics world. The Movie Toolbox derives the display boundary region only from enabled tracks, and only from those tracks that are used in the current display mode (that is, movie, poster, or preview). The display boundary region is valid for the current movie time.

Special Considerations

Your application must dispose of the returned handle when it is done with it.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
Inside Mac Movie TB Code

Declared In
Movies.h

GetMovieDisplayClipRgn

Determines a movie's current display clipping region.

```
RgnHandle GetMovieDisplayClipRgn (
   Movie theMovie
);
```

Parameters

theMovie

The movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie` (page 108), `NewMovieFromFile`, and `NewMovieFromHandle`.

Return Value

A handle to a `MacRegion` structure that the function allocates. If the movie does not have a spatial representation at the current time, the function returns an empty region. If the function could not satisfy the request, it sets the returned handle to `NIL`.

Special Considerations

Your application must dispose of the returned handle when it is done with it. Note that the display clipping region is not saved with the movie.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Movies.h`

GetMovieDuration

Returns the duration of a movie.

```
TimeValue GetMovieDuration (
   Movie theMovie
);
```

Parameters

theMovie

The movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie` (page 108), `NewMovieFromFile`, and `NewMovieFromHandle`.

Return Value

The duration of the designated movie.

Discussion

This function returns a time value, expressed in the movie's time scale, that is calculated to be the maximum durations of all the tracks in the movie. The following code sample illustrates its use:

```
// GetMovieDuration coding example
// See "Discovering QuickTime," page 363
Movie           movie1;
TimeValue       lOldDuration;
Movie           movie2;
long            lIndex, lOrigTrackCount, lReferenceIndex;
Track           track, trackSprite;
// get the first track in original movie and position at the start
trackSprite =GetMovieIndTrack(movie1, 1);
```

```
SetMovieSelection(movie1, 0, 0);
// remove all tracks except video in modifier movie
for (lIndex =1; lIndex <=GetMovieTrackCount(movie2); lIndex++) {
    Track       track =GetMovieIndTrack(movie2, lIndex);
    OSType      dwType;
    GetMediaHandlerDescription(GetTrackMedia(track),
                                &dwType, NIL, NIL);
    if (dwType !=VideoMediaType) {
        DisposeMovieTrack(track);
        lIndex--;
    }
}
// add the modifier track to original movie
lOldDuration =GetMovieDuration(movie1);
AddMovieSelection(movie1, movie2);
DisposeMovie(movie2);
// truncate the movie to the length of the original track
DeleteMovieSegment(movie1, lOldDuration,
                   GetMovieDuration(movie1) - lOldDuration);
// associate the modifier track with the original sprite track
track =GetMovieIndTrack(movie1, lOrigTrackCount + 1);
AddTrackReference(trackSprite, track, kTrackModifierReference,
                   &lReferenceIndex);
```

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
CompressMovies
MovieGWorlds
QT Internals
qtstreamsplicer
qtstreamsplicer.win

Declared In
`Movies.h`

GetMovieGWorld

Returns a movie's graphics world.

```
void GetMovieGWorld (
   Movie theMovie,
   CGrafPtr *port,
   GDHandle *gdh
);
```

Parameters

`theMovie`

The movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie` (page 108), `NewMovieFromFile`, and `NewMovieFromHandle`.

port

A pointer to a field that is to receive a pointer to a `CGrafPort` structure. Set this parameter to `NIL` if you don't want this information.

gdh

A pointer to a field that is to receive a handle to a `GDevice` structure. Set this parameter to `NIL` if you don't want this information.

Return Value
You can access this function's error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71).

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
BurntTextSampleCode
Inside Mac Movie TB Code
MovieGWorlds
vrscript
vrscript.win

Declared In
`Movies.h`

GetMovieMatrix

Retrieves a movie's transformation matrix.

```
void GetMovieMatrix (
   Movie theMovie,
   MatrixRecord *matrix
);
```

Parameters

theMovie

The movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie` (page 108), `NewMovieFromFile`, and `NewMovieFromHandle`.

matrix

A pointer to a `MatrixRecord` structure, where `GetMovieMatrix` returns the movie's matrix.

Return Value
You can access this function's error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71).

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
BurntTextSampleCode
QTCarbonShell
qtskins.win
vrmovies.win
vrscript

Declared In
`Movies.h`

GetMovieModificationTime

Returns a movie's modification date and time.

```
unsigned long GetMovieModificationTime (
   Movie theMovie
);
```

Parameters

theMovie

The movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie` (page 108), `NewMovieFromFile`, and `NewMovieFromHandle`.

Return Value
The movie's modification date and time.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Movies.h`

GetMovieNaturalBoundsRect

Gets a movie's natural boundary rectangle.

```
void GetMovieNaturalBoundsRect (
   Movie theMovie,
   Rect *naturalBounds
);
```

Parameters

theMovie

A movie identifier. Your application obtains this identifier from such functions as `NewMovie` (page 108), `NewMovieFromFile`, and `NewMovieFromHandle`.

naturalBounds

A pointer to a `Rect` structure that represents the movie's bounding rectangle.

Return Value
You can access this function's error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71).

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
CarbonQTGraphicImport
QTCarbonShell
SimpleVideoOut
ThreadsExportMovie
ThreadsImportMovie

Declared In
`Movies.h`

GetMoviePict

Creates a QuickDraw picture from a specified movie at a specified time.

```
PicHandle GetMoviePict (
   Movie theMovie,
   TimeValue time
);
```

Parameters

theMovie

The movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie` (page 108), `NewMovieFromFile`, and `NewMovieFromHandle`.

time

The movie time from which the image is to be taken.

Return Value
A handle to a `Picture` structure. If the function could not create the picture, the returned handle is set to `NIL`.

Discussion
This function uses only those movie tracks that are currently enabled and would therefore be used in playback. Your application may call this function even if the movie is inactive.

Special Considerations

Your application must dispose of this picture handle.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
AlwaysPreview
DigitizerShell
DragAndDrop Shell
MovieGWorlds
VelEng Wavelet

Declared In
`Movies.h`

GetMoviePosterPict

Creates a QuickDraw picture that contains a movie's poster.

```
PicHandle GetMoviePosterPict (
   Movie theMovie
);
```

Parameters

`theMovie`

The movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie` (page 108), `NewMovieFromFile`, and `NewMovieFromHandle`.

Return Value
A handle to a `Picture` structure. If the function could not create the picture, the returned handle is set to `NIL`.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
bMoviePaletteCocoa
CompressMovies
MovieGWorlds
qtcompress.win
qtinfo

Declared In
`Movies.h`

GetMoviePosterTime

Returns the poster's time in a movie.

```
TimeValue GetMoviePosterTime (
   Movie theMovie
);
```

Parameters

theMovie

The movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie` (page 108), `NewMovieFromFile`, and `NewMovieFromHandle`.

Return Value
The time in the movie from which its poster is taken.

Discussion
Since a movie poster has no duration, it is defined by a point in time within the movie. The time value returned by `GetMoviePosterTime` is in the time coordinate system of the movie and represents the starting time for the movie frame that contains the poster image.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
MakeEffectMovie
qteffects
qteffects.win
samplemakeeffectmovie
samplemakeeffectmovie.win

Declared In
`Movies.h`

GetMoviePreferredRate

Returns a movie's default playback rate.

```
Fixed GetMoviePreferredRate (
   Movie theMovie
);
```

Parameters

theMovie

The movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie` (page 108), `NewMovieFromFile`, and `NewMovieFromHandle`.

Return Value
The movie's default playback rate.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
MovieGWorlds
qtbigscreen
qtbigscreen.win
vrscript
vrscript.win

Declared In
`Movies.h`

GetMoviePreferredVolume

Returns a movie's preferred volume setting.

```
short GetMoviePreferredVolume (
   Movie theMovie
);
```

Parameters

`theMovie`

The movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie` (page 108), `NewMovieFromFile`, and `NewMovieFromHandle`.

Return Value
The movie's preferred volume setting.

Discussion
A movie's tracks have their own volume settings. A track's volume is scaled by the movie's volume to produce the track's final volume. On Macintosh computers, the movie's volume is further scaled by the sound volume that the user controls from the Sound control panel.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
MakeEffectMovie
qteffects.win
qtgraphics.win
qtwiredactions
vrbackbuffer.win

Declared In
`Movies.h`

GetMoviePreviewMode

Determines whether a movie is in preview mode.

```
Boolean GetMoviePreviewMode (
   Movie theMovie
);
```

Parameters

theMovie

The movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie` (page 108), `NewMovieFromFile`, and `NewMovieFromHandle`.

Return Value
TRUE if the movie is in preview mode; FALSE if the movie is in normal playback mode.

Discussion
If a movie is in preview mode, only the movie's preview can be displayed.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Movies.h`

GetMoviePreviewTime

Returns the starting time and duration of the movie's preview.

```
void GetMoviePreviewTime (
   Movie theMovie,
   TimeValue *previewTime,
   TimeValue *previewDuration
);
```

Parameters

theMovie

The movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie` (page 108), `NewMovieFromFile`, and `NewMovieFromHandle`.

previewTime

A pointer to a time value. The Movie Toolbox places the preview's starting time into the field referred to by this parameter. If the movie does not have a preview, the Movie Toolbox sets this returned value to 0.

previewDuration

A pointer to a time value. The Movie Toolbox places the preview's duration into the field referred to by this parameter. If the movie does not have a preview, the Movie Toolbox sets this returned value to 0.

Return Value
You can access this function's error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71).

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
qtinfo
qtinfo.win

Declared In
`Movies.h`

GetMovieRate

Returns a movie's playback rate.

```
Fixed GetMovieRate (
   Movie theMovie
);
```

Parameters

`theMovie`

The movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie` (page 108), `NewMovieFromFile`, and `NewMovieFromHandle`.

Return Value
The rate at which the movie is currently playing, expressed as a 32-bit fixed-point number. Positive integers indicate forward rates and negative integers indicate reverse rates. A value of 1 indicates normal speed, a value of 2 indicates double speed, -2 means the movie is playing backward at double speed, and so on. A value of 0 means the movie is paused or stopped.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
QT Internals
QTCarbonShell
vrscript
vrscript.win

Declared In
`Movies.h`

GetMovieRateChangeConstraints

Returns the minimum and maximum delay you can get when a movie's rate is changed.

```
OSErr GetMovieRateChangeConstraints (
   Movie theMovie,
   TimeRecord *minimumDelay,
   TimeRecord *maximumDelay
);
```

Parameters

theMovie

The movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie`, `NewMovieFromFile`, and `NewMovieFromHandle`.

minimumDelay

A pointer to a `TimeRecord` structure. The function updates this structure to contain the minimum delay when a rate change happens.

maximumDelay

A pointer to a `TimeRecord` structure. The function updates this structure to contain the maximum delay when a rate change happens.

Discussion

If the time base master clock of the movie is changed, this function must be called again to reflect the current constraints.

Version Notes

Introduced in QuickTime 6.4.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`Movies.h`

GetMovieSelection

Returns information about a movie's current selection.

```
void GetMovieSelection (
   Movie theMovie,
   TimeValue *selectionTime,
   TimeValue *selectionDuration
);
```

Parameters

theMovie

The movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie` (page 108), `NewMovieFromFile`, and `NewMovieFromHandle`.

selectionTime

A pointer to a time value. The `GetMovieSelection` function places the starting time of the current selection into the field referred to by this parameter. Set this parameter to `NIL` if you don't want this information.

selectionDuration

A pointer to a time value. The `GetMovieSelection` function places the duration of the current selection into the field referred to by this parameter. Set this parameter to `NIL` if you don't want this information.

Return Value
You can access this function's error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71).

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
qtinfo
qtinfo.win

Declared In
`Movies.h`

GetMoviesError

Returns the contents of the current error value and resets the current error value to 0.

```
OSErr GetMoviesError (
   void
);
```

Return Value
See `Error Codes`. Returns `noErr` if there is no error in the current error value.

Discussion
The Movie Toolbox provides two error values to your application: the current error and the sticky error. The current error is the result code from the last Movie Toolbox function; it is updated each time your application calls a Movie Toolbox function. The following code sample shows a typical use:

```
// GetMoviesError coding example
// See "Discovering QuickTime," page 256
OSErr QTUtils_SaveMovie (Movie theMovie)
{
    StandardFileReply   mySFReply;
    StringPtr   myPrompt =QTUtils_ConvertCToPascalString(kSavePrompt);
    StringPtr   myFileName =
                    QTUtils_ConvertCToPascalString(kSaveMovieFileName);
    OSErr       myErr =noErr;
    if (theMovie ==NIL)
        return(invalidMovie);
    StandardPutFile(myPrompt, myFileName, &mySFReply);
    if (mySFReply.sfGood) {
        FlattenMovieData(   theMovie,
                            flattenAddMovieToDataFork,
                            &mySFReply.sfFile,
                            FOUR_CHAR_CODE('TVOD'),
                            smSystemScript,
                            createMovieFileDeleteCurFile);
        myErr =GetMoviesError();
    }
    free(myPrompt);
    free(myFileName);
```

```
    return(myErr);
}
```

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
CompressMovies
DigitizerShell
DragAndDrop Shell
MovieGWorlds
QT Internals

Declared In
`Movies.h`

GetMoviesStickyError

Returns the contents of the sticky error value.

```
OSErr GetMoviesStickyError (
   void
);
```

Return Value
See `Error Codes`. Returns `noErr` if there is no error in the sticky error value.

Discussion
The sticky error value contains the first nonzero result code from any Movie Toolbox function that you called after having cleared the sticky error with `ClearMoviesStickyError` (page 30).

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
BurntTextSampleCode
vrmakeobject
vrmakepano
VRMakePano Library
vrmakepano.win

Declared In
`Movies.h`

GetMovieTime

Returns a movie's current time both as a time value and in a time structure.

```
TimeValue GetMovieTime (
   Movie theMovie,
   TimeRecord *currentTime
);
```

Parameters

theMovie

The movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie` (page 108), `NewMovieFromFile`, and `NewMovieFromHandle`.

currentTime

A pointer to a `TimeRecord` structure. The function updates this time structure to contain the movie's current time. If you don't want this information, set this parameter to `NIL`.

Return Value

The time value of the current time.

Discussion

This function returns the movie's current time value in two formats: as a time value and in a time structure.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

CompressMovies

DragAndDrop Shell

MovieBrowser

MovieGWorlds

QT Internals

Declared In

`Movies.h`

GetMovieTimeBase

Returns a movie's time base.

```
TimeBase GetMovieTimeBase (
   Movie theMovie
);
```

Parameters

theMovie

The movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie` (page 108), `NewMovieFromFile`, and `NewMovieFromHandle`.

Return Value

The movie's `TimeBaseRecord` structure.

Special Considerations

The Movie Toolbox disposes of a movie's time base when you dispose of the movie.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
LiveVideoMixer2
qtinfo
vrmovies
vrscript
vrscript.win

Declared In
`Movies.h`

GetMovieTimeScale

Returns the time scale of a movie.

```
TimeScale GetMovieTimeScale (
   Movie theMovie
);
```

Parameters

`theMovie`

The movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie` (page 108), `NewMovieFromFile`, and `NewMovieFromHandle`.

Return Value
A long integer that contains the movie's time scale.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
MakeEffectMovie
qteffects
qteffects.win
qtstreamsplicer
qtstreamsplicer.win

Declared In
`Movies.h`

GetMovieUserData

Obtains access to a movie's user data list.

```
UserData GetMovieUserData (
   Movie theMovie
);
```

Parameters

theMovie

The movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie` (page 108), `NewMovieFromFile`, and `NewMovieFromHandle`.

Return Value

The `UserDataRecord` structure for the movie. If the function could not locate the movie's user data, it sets this return value to `NIL`.

Discussion

This function returns a reference to the movie's user data list, which is valid until you dispose of the movie. When you save the movie, the Movie Toolbox saves the user data as well.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

MakeEffectMovie
qtactiontargets
qtactiontargets.win
qtwiredactions
qtwiredactions.win

Declared In

`Movies.h`

GetMovieVisualContext

Returns the current visual context for a movie.

```
OSStatus GetMovieVisualContext (
   Movie movie,
   QTVisualContextRef *visualContext
);
```

Parameters

movie

The movie.

visualContext

A pointer to a variable to receive the visual context.

Return Value
An error code. Returns `noErr` if there is no error. Returns `memFullErr` if memory cannot be allocated. Returns `kQTVisualContextRequiredErr` if the movie is not using a visual context. Returns `paramErr` if the movie or `visualContextOut` is NULL.

Discussion
Returns the `QTVisualContext` object associated with the movie. You are responsible for retaining and releasing the object as needed (that is, if the returned object has not been retained for you). If the visual context was set to NULL (see `SetMovieVisualContext` (page 151)), `noErr` is returned and `visualContextOut` receives NULL.

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

Declared In
`Movies.h`

GetMovieVolume

Returns a movie's current volume setting.

```
short GetMovieVolume (
   Movie theMovie
);
```

Parameters

theMovie
The movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie` (page 108), `NewMovieFromFile`, and `NewMovieFromHandle`.

Return Value
The current volume setting for the movie.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
MakeEffectMovie
qteffects.win
qtgraphics.win
vrscript
vrscript.win

Declared In
`Movies.h`

GetNextCallBack

Returns the next callback event associated with a specified time base.

```
QTCallBack GetNextCallBack (
   QTCallBack cb
);
```

Parameters

cb

Specifies the starting callback event for the operation. Your clock component obtains this value from the `GetFirstCallBack` (page 52) function or from previous calls to the `GetNextCallBack` function.

Return Value

A pointer to a `CallBackRecord` structure. Your software can pass this structure to other functions, such as `ClockRateChanged`.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Movies.h`

GetNextTrackForCompositing

Determines the next track in a movie's compositing process.

```
Track GetNextTrackForCompositing (
   Movie theMovie,
   Track theTrack
);
```

Parameters

theMovie

A movie identifier. Your application obtains this identifier from such functions as `NewMovie` (page 108), `NewMovieFromFile`, and `NewMovieFromHandle`.

theTrack

The identifier of the track from which to start.

Return Value

The returned identifier of the next track to be composited.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Movies.h`

GetPrevTrackForCompositing

Determines the previous track in a movie's compositing process.

```
Track GetPrevTrackForCompositing (
   Movie theMovie,
   Track theTrack
);
```

Parameters

theMovie

A movie identifier. Your application obtains this identifier from such functions as NewMovie (page 108), NewMovieFromFile, and NewMovieFromHandle.

theTrack

The identifier of the track from which to start.

Return Value
The returned identifier of the previous track in the compositing process.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
Movies.h

GetTimeBaseEffectiveRate

Returns the effective rate at which the specified time base is moving relative to its master clock.

```
Fixed GetTimeBaseEffectiveRate (
   TimeBase tb
);
```

Parameters

tb

The time base for this operation. Your application obtains this time base identifier from the NewTimeBase (page 110) function.

Return Value
The effective rate at which the time base specified by tb is moving relative to its master clock.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
Movies.h

GetTimeBaseFlags

Obtains the control flags of a time base.

```
long GetTimeBaseFlags (
   TimeBase tb
);
```

Parameters

tb

The time base for this operation. Your application obtains this time base identifier from `NewTimeBase` (page 110).

Return Value
Control flags (see below). Unused flags are set to 0.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
MakeEffectMovie
vrmovies
vrmovies.win
vrscript
vrscript.win

Declared In
`Movies.h`

GetTimeBaseMasterClock

Determines the clock component that is assigned to a time base.

```
ComponentInstance GetTimeBaseMasterClock (
   TimeBase tb
);
```

Parameters

tb

The time base for this operation. Your application obtains this time base identifier from the `NewTimeBase` (page 110) function.

Return Value
A reference to a component instance. If a clock component is not assigned to the time base, the returned reference is `NIL`. In this case, the time base relies on another time base for its time source. Use `GetTimeBaseMasterTimeBase` (page 79) to obtain the time base reference to that master time base.

Discussion
This function returns a reference to a component instance of the clock component that provides a time source to the specified time base. Every time base derives its time from either a clock component or from another time base. If a time base derives its time from a clock component, use this function to obtain the component instance of the clock component.

Special Considerations

The Component Manager allows a single component to serve multiple client applications at the same time. Each client application has a unique connection to the component, identified by a component instance. Don't close this connection; the time base is using it to maintain its time source.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
BrideOfMungGrab
QTMusicToo

Declared In
`Movies.h`

GetTimeBaseMasterOffsetTimeBase

Allows an offset time base to retrieve the master time base it is attached to.

```
TimeBase GetTimeBaseMasterOffsetTimeBase (
   TimeBase tb
);
```

Parameters

tb
An offset time base.

Return Value
The master time base for the offset time base passed in `tb`. Returns `NIL` if `tb` does not contain an offset time base.

Version Notes
Introduced in QuickTime 6.4.

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

Declared In
`Movies.h`

GetTimeBaseMasterTimeBase

Determines the master time base that is assigned to a time base.

```
TimeBase GetTimeBaseMasterTimeBase (
   TimeBase tb
);
```

Parameters

tb

The time base for this operation. Your application obtains this time base identifier from `NewTimeBase` (page 110).

Return Value

A time base. If a master time base is not assigned to the time base, this function sets the returned reference to `NIL`. In this case, the time base relies on a clock component for its time source. Use `GetTimeBaseMasterClock` (page 78) to obtain the component instance reference to that clock component.

Discussion

This function returns a reference to the master time base that provides a time source to this time base. A time base derives its time from either a clock component or from another time base. If a time base derives its time from another time base, use this function to obtain the identifier for that master time base.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Movies.h`

GetTimeBaseRate

Retrieves the rate of a time base.

```
Fixed GetTimeBaseRate (
   TimeBase tb
);
```

Parameters

tb

The time base for this operation. Your application obtains this time base identifier from the `NewTimeBase` (page 110) function.

Return Value

The time base's rate. This rate value may be nonzero even if the time base has stopped, because it has reached its stop time. Rates may be set to negative values, which cause time to move backward for the time base.

Discussion

This function returns the current rate of the time base as a fixed-point number.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

SoftVDigX

Declared In
Movies.h

GetTimeBaseRateChangeStatus

Lets a time base client determine the time base's last rate change status.

```
OSErr GetTimeBaseRateChangeStatus (
   TimeBase tb,
   TimeScale scale,
   Fixed *ratedChangedTo,
   TimeBaseStatus *flags,
   TimeRecord *rateChangeTimeBaseTime,
   TimeRecord *rateChangeClockTime,
   TimeRecord *currentClockTime
);
```

Parameters

tb

A pointer to a TimeBaseRecord structure.

scale

The scale to use for the returned time values. Pass 0 to retrieve the time in the preferred time scale of the time base.

rateChangedTo

The rate value changed to. Clients may pass NIL if they do not want to receive this information.

flags

A pointer to a flag (see below) that will be returned when the clock is waiting for a future time to start moving while its rate is nonzero. When set, the unpinned time will return a negative value telling how far you are from the real start time. Clients may pass NIL if they do not want to receive this information. rateChangeTimeBaseTime The time base time when the rate changed. Clients may pass NIL if they do not want to receive this information. rateChangeClockTime The clock time when the rate changed. Clients may pass NIL if they do not want to receive this information. currentClockTime The current clock time value. Clients may pass NIL if they do not want to receive this information. timeBaseRateChanging The clock is waiting for a future time to start moving while its rate is nonzero. When set, the unpinned time will return a negative value telling how far you are from the real start time. See these constants:

```
timeBaseRateChanging
```

rateChangeTimeBaseTime

The time base time when the rate changed. Clients may pass NIL if they do not want to receive this information.

rateChangeClockTime

The clock time when the rate changed. Clients may pass NIL if they do not want to receive this information.

currentClockTime

The current clock time value. Clients may pass NIL if they do not want to receive this information.

Discussion
When the flag timeBaseRateChanging is returned, the amount of time left before the time base ticks is equal to (rateChangeClockTime - currentClockTime).

Version Notes
Introduced in QuickTime 6.4.

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

Declared In
`Movies.h`

GetTimeBaseStartTime

Determines the start time of a time base.

```
TimeValue GetTimeBaseStartTime (
   TimeBase tb,
   TimeScale s,
   TimeRecord *tr
);
```

Parameters

tb

The time base for this operation. Your application obtains this time base identifier from the `NewTimeBase` (page 110) function.

s

The time scale in which to return the start time.

tr

A pointer to a time structure that is to receive the start time. This is an optional parameter. If you don't want the time value represented in a time structure, set this parameter to `NIL`.

Return Value
The time base's start time.

Discussion
This function returns a time value that contains the start time for the specified time base in the specified time scale. The function returns this value even if you specify a time structure with the `tr` parameter.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Movies.h`

GetTimeBaseStatus

Determines when the current time of a time base would fall outside of the range of values specified by the time base's start and stop times.

```
long GetTimeBaseStatus (
   TimeBase tb,
   TimeRecord *unpinnedTime
);
```

Parameters

tb

The time base for this operation. Your application obtains this time base identifier from the `NewTimeBase` (page 110) function.

unpinnedTime

A pointer to a time structure that is to receive the current time of the time base. Note that this time value may be outside the range of values specified by the start and stop times of the time base.

Return Value
Status flags (see below).

Discussion
The status information returned by this function allows you to determine when the current time of a time base would fall outside of the range of values specified by the start and stop times of the time base. This can happen when a time base relies on a master time base or when its time has reached the stop time.

Special Considerations

This function returns no error codes.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Movies.h`

GetTimeBaseStopTime

Determines the stop time of a time base.

```
TimeValue GetTimeBaseStopTime (
   TimeBase tb,
   TimeScale s,
   TimeRecord *tr
);
```

Parameters

tb

The time base for this operation. Your application obtains this time base identifier from the `NewTimeBase` (page 110) function.

s

The time scale in which to return the stop time.

tr

A pointer to a time structure that is to receive the stop time. This is an optional parameter. If you don't want the time value represented in a time structure, set this parameter to `NIL`.

Return Value
The time base's stop time.

Discussion
This function returns a time value that contains the stop time for the specified time base in the specified time scale. The function returns this value even if you specify a time structure with the `tr` parameter.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Movies.h`

GetTimeBaseThreadAttachState

Determines whether a given time base is attached to a thread.

```
OSErr GetTimeBaseThreadAttachState (
   TimeBase inTimeBase,
   Boolean *outAttachedToCurrentThread,
   Boolean *outAttachedToAnyThread
);
```

Parameters

`inTimeBase`
A time base.

`outAttachedToCurrentThread`
A pointer to a Boolean that on exit is TRUE if the time base is attached to the current thread, FALSE otherwise.

`outAttachedToAnyThread`
A pointer to a Boolean that on exit is TRUE if the time base is attached to any thread, FALSE otherwise.

Return Value
See `Error Codes` in the QuickTime API Reference. Returns `noErr` if there is no error.

Version Notes
Introduced in QuickTime 6.4.

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

Declared In
`Movies.h`

GetTimeBaseTime

Obtains the current time value from a time base.

```
TimeValue GetTimeBaseTime (
   TimeBase tb,
   TimeScale s,
   TimeRecord *tr
);
```

Parameters

tb

The time base for this operation. Your application obtains this time base identifier from the `NewTimeBase` (page 110) function.

s

The time scale in which to return the current time value. Set this parameter to 0 to retrieve the time in the preferred time scale of the time base.

tr

A pointer to a time structure that is to receive the current time value. This is an optional parameter. If you don't want the time value represented in a time structure, set this parameter to `NIL`.

Return Value
The time base's current time.

Discussion
This function returns a time value that contains the current time from the specified time base in the specified time scale. The function returns this value even if you specify a time structure with the `tr` parameter.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
AlwaysPreview
BrideOfMungGrab
QTMusicToo
SoftVDigX

Declared In
`Movies.h`

GetTrackBoundsRgn

Lets the media limit the size of a track boundary rectangle.

```
RgnHandle GetTrackBoundsRgn (
   Track theTrack
);
```

Parameters

theTrack

The track for this operation. Your application obtains this track identifier from such functions as `NewMovieTrack` and `GetMovieTrack`.

Return Value
A handle to the region limited by the media.

Discussion
Because the media limits the size of the track boundary rectangle, the region returned by `GetTrackBoundsRgn` may not be rectangular and may be smaller than the track boundary region.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Movies.h`

GetTrackClipRgn

Determines the clipping region of a track.

```
RgnHandle GetTrackClipRgn (
   Track theTrack
);
```

Parameters

`theTrack`
The track for this operation. Your application obtains this track identifier from such functions as `NewMovieTrack` and `GetMovieTrack`.

Return Value
A handle to the track's clipping region.

Discussion
This function allocates the region and returns a handle to the region. Your application must dispose of this region when you are done with it. If the function could not satisfy your request or if there is no clipping region defined for the track, it sets the returned handle to `NIL`.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Movies.h`

GetTrackDisplayBoundsRgn

Determines the region a track occupies in a movie's graphics world.

```
RgnHandle GetTrackDisplayBoundsRgn (
   Track theTrack
);
```

Parameters

theTrack

The track for this operation. Your application obtains this track identifier from such functions as `NewMovieTrack` and `GetMovieTrack`.

Return Value
A handle to the region the specified track occupies in a movie's graphics world.

Discussion
This function allocates the region and returns a handle to the region. If the track does not have a spatial representation at the current movie time, the function returns an empty region. If the function could not satisfy your request, it sets the returned handle to `NIL`.

Special Considerations

Your application must dispose of the returned region when you are done with it.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
BurntTextSampleCode

Declared In
`Movies.h`

GetTrackMatte

Retrieves a copy of a track's matte.

```
PixMapHandle GetTrackMatte (
   Track theTrack
);
```

Parameters

theTrack

The track for this operation. Your application obtains this track identifier from such functions as `NewMovieTrack` and `GetMovieTrack`.

Return Value
A handle to a `PixMap` structure that represents the specified track's matte. If the function could not satisfy your request, it sets the returned handle to `NIL`.

Discussion
The matte defines which of the track's pixels are displayed in a movie, and it is valid for the entire duration of the movie.

Special Considerations

You should use `DisposeMatte` (page 35) to dispose of the matte when you are finished with it.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
Inside Mac Movie TB Code

Declared In
`Movies.h`

GetTrackMovieBoundsRgn

Determines the region the track occupies in a movie's boundary region.

```
RgnHandle GetTrackMovieBoundsRgn (
   Track theTrack
);
```

Parameters

`theTrack`

The track for this operation. Your application obtains this track identifier from such functions as `NewMovieTrack` and `GetMovieTrack`.

Return Value
A handle to the region the specified track occupies in its movie's boundary region. If the track does not have a spatial representation at the current movie time, the function returns an empty region. If the function could not satisfy your request, it sets the returned handle to `NIL`.

Discussion
This function determines the region by applying the track's clipping region and matrix. This region is valid only for the current movie time. The function allocates the region and returns a handle to it.

Special Considerations

Your application must dispose of the returned region when you are done with it.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Movies.h`

GetTrackPict

Creates a QuickDraw picture from a specified track at a specified time.

```
PicHandle GetTrackPict (
   Track theTrack,
   TimeValue time
);
```

Parameters

theTrack

The track for this operation. Your application obtains this track identifier from such functions as `NewMovieTrack` and `GetMovieTrack`.

time

The time at which the image is taken.

Return Value

A handle to the specified picture. If the function could not create the picture, the returned handle is set to `NIL`.

Special Considerations

Your application must dispose of the returned picture handle.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

MakeEffectMovie

qteffects

qteffects.win

samplemakeeffectmovie

samplemakeeffectmovie.win

Declared In

`Movies.h`

GoToBeginningOfMovie

Repositions a movie to play from its start.

```
void GoToBeginningOfMovie (
   Movie theMovie
);
```

Parameters

theMovie

The movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie` (page 108), `NewMovieFromFile`, and `NewMovieFromHandle`.

Discussion

If the movie is in preview mode, the function goes to the start of the preview segment of the movie. In all other cases, this function goes to the start of the movie, where the movie time value is 0. You can access error returns from this function through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71). See `Error Codes`.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
Graphic Import-Export
MovieGWorlds
OpenGLCompositorLab
vrscript
vrscript.win

Declared In
`Movies.h`

GoToEndOfMovie

Repositions a movie to play from its end.

```
void GoToEndOfMovie (
   Movie theMovie
);
```

Parameters

theMovie

The movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie` (page 108), `NewMovieFromFile`, and `NewMovieFromHandle`.

Return Value
You can access this function's error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71).

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Movies.h`

InvalidateMovieRegion

Invalidates a small area of a movie.

```
OSErr InvalidateMovieRegion (
   Movie theMovie,
   RgnHandle invalidRgn
);
```

Parameters

theMovie

The movie whose area you wish to invalidate. Your application obtains this movie identifier from such functions as NewMovie (page 108), NewMovieFromFile, and NewMovieFromHandle.

invalidRgn

A region indicating the area of the movie to invalidate. If necessary, QuickTime will make a copy of this region. To invalidate the entire movie area, pass NIL for this parameter.

Return Value

You can access Movie Toolbox error returns through GetMoviesError (page 70) and GetMoviesStickyError (page 71), as well as in the function result. See Error Codes.

Discussion

Use this function instead of UpdateMovie (page 196) to invalidate a small area of a movie. It marks all areas of the movie that intersect the invalidRgn parameter. The next time you call MoviesTask (page 106), the Movie Toolbox redraws the marked areas. This provides a way to invalidate a portion of the movie's area instead of its entire area, as does UpdateMovie. This allows for higher performance update handling when a movie has many tracks or covers a large area. For handling of update events, applications should continue to use UpdateMovie.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

Movies.h

IsMovieDone

Determines if a particular movie has completely finished playing.

```
Boolean IsMovieDone (
   Movie theMovie
);
```

Parameters

theMovie

The movie for this operation. Your application obtains this movie identifier from such functions as NewMovie (page 108), NewMovieFromFile, and NewMovieFromHandle.

Return Value

Returns TRUE if the specified movie has finished playing, otherwise returns FALSE.

Discussion
A movie with a positive rate (playing forward) is considered done when its movie time reaches the movie end time. Conversely, a movie with a negative rate (playing backward) is considered done when its movie time reaches the movie start time. If your application has changed the movie's active segment, the status returned by this function is relative to the active segment, rather than to the entire movie. You can use `SetMovieActiveSegment` (page 134) to change a movie's active segment.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
Graphic Import-Export
MovieGWorlds
SimpleCocoaMovie
vrscript
vrscript.win

Declared In
`Movies.h`

ITextAddString

Undocumented

```
OSErr ITextAddString (
   QTAtomContainer container,
   QTAtom parentAtom,
   RegionCode theRegionCode,
   ConstStr255Param theString
);
```

Parameters

container
Undocumented

parentAtom
Undocumented

theRegionCode
A 16-bit signed integer containing an international region code; see `Localization Codes`.

theString
The string to add.

Return Value
You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
Movies.h

lTextGetString

Undocumented

```
OSErr lTextGetString (
   QTAtomContainer container,
   QTAtom parentAtom,
   RegionCode requestedRegion,
   RegionCode *foundRegion,
   StringPtr theString
);
```

Parameters

container
Undocumented

parentAtom
Undocumented

requestedRegion
Undocumented

foundRegion
On return, a 16-bit signed integer containing an international region code; see Localization Codes.

theString
The found string.

Return Value
You can access Movie Toolbox error returns through GetMoviesError (page 70) and GetMoviesStickyError (page 71), as well as in the function result. See Error Codes.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
Movies.h

lTextRemoveString

Undocumented

```
OSErr ITextRemoveString (
   QTAtomContainer container,
   QTAtom parentAtom,
   RegionCode theRegionCode,
   long flags
);
```

Parameters

container

Undocumented

parentAtom

Undocumented

theRegionCode

A 16-bit signed integer containing an international region code; see `Localization Codes`.

flags

Flags (see below) that modify the process. See these constants:

`kITextRemoveEverythingBut`

`kITextRemoveLeaveSuggestedAlternate`

Return Value

You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Movies.h`

LoadMediaIntoRam

Loads a media's data into memory.

```
OSErr LoadMediaIntoRam (
   Media theMedia,
   TimeValue time,
   TimeValue duration,
   long flags
);
```

Parameters

theMedia

The media for this operation. Your application obtains this media identifier from such functions as `NewTrackMedia` and `GetTrackMedia`. See `Media Identifiers`.

time

The starting time of the media segment to load. This time value must be expressed in the media's time coordinate system.

duration

The length of the segment to load. Use `GetMediaDuration` to determine the length of the entire media. Note that the media handler may load more data than you specify if the media data was added in larger pieces.

flags

Flags that give you explicit control over what is loaded into memory and how long to keep it around. See `RAM Loading Flags`. You can set these flags in any combination.

Return Value

You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Discussion

The exact behavior of `LoadMediaIntoRam` is dependent on the media handler.

Special Considerations

If `LoadMediaIntoRam` fails because it is out of memory, no data is purged.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Movies.h`

LoadMovieIntoRam

Loads a movie's data into memory.

```
OSErr LoadMovieIntoRam (
   Movie theMovie,
   TimeValue time,
   TimeValue duration,
   long flags
);
```

Parameters

theMovie

The movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie` (page 108), `NewMovieFromFile`, and `NewMovieFromHandle`.

time

The starting time of the movie segment to load.

duration

The length of the segment to load. Use `GetMovieDuration` (page 59) to determine the length of the entire movie. Note that the Movie Toolbox may load more data than you specify due to the way the data is loaded.

flags

Flags that give you explicit control over what is loaded into memory and how long to keep it around. See `RAM Loading Flags`. You can set these flags in any combination.

Return Value
You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
CompressMovies
DigitizerShell
DragAndDrop Shell
MovieGWorlds
QT Internals

Declared In
`Movies.h`

LoadTrackIntoRam

Loads a track's data into memory.

```
OSErr LoadTrackIntoRam (
   Track theTrack,
   TimeValue time,
   TimeValue duration,
   long flags
);
```

Parameters

theTrack
The track for this operation. Your application obtains this track identifier from such functions as `NewMovieTrack` and `GetMovieTrack`.

time
The starting time of the track segment to load. You must specify this time value in the movie's time coordinate system.

duration
The length of the segment to load. Use `GetTrackDuration` to determine the length of the entire movie. Note that the media handler may load more data than you specify.

flags
Flags that give you explicit control over what is loaded into memory and how long to keep it around. See `RAM Loading Flags`. You can set these flags in any combination.

Return Value
If the track does not fit, the function returns an error. See `Error Codes`. Returns `noErr` if there is no error.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
qttext
qttext.win

Declared In
Movies.h

Media3DGetCameraAngleAspect

Deprecated.

```
ComponentResult Media3DGetCameraAngleAspect (
   MediaHandler mh,
   QTFloatSingle *fov,
   QTFloatSingle *aspectRatioXToY
);
```

Version Notes
Introduced in QuickTime 4.

Availability
Available in Mac OS X v10.0 and later.

Declared In
Movies.h

Media3DGetCameraData

Deprecated.

```
ComponentResult Media3DGetCameraData (
   MediaHandler mh,
   void *cameraData
);
```

Version Notes
Introduced in QuickTime 4.

Availability
Available in Mac OS X v10.0 and later.

Declared In
Movies.h

Media3DGetCameraRange

Deprecated.

```
ComponentResult Media3DGetCameraRange (
   MediaHandler mh,
   void *tQ3CameraRange
);
```

Version Notes
Introduced in QuickTime 4.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Movies.h`

Media3DGetCurrentGroup

Deprecated.

```
ComponentResult Media3DGetCurrentGroup (
   MediaHandler mh,
   void *group
);
```

Version Notes
Introduced in QuickTime 4.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Movies.h`

Media3DGetNamedObjectList

Deprecated.

```
ComponentResult Media3DGetNamedObjectList (
   MediaHandler mh,
   QTAtomContainer *objectList
);
```

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Movies.h`

Media3DGetRendererList

Deprecated.

```
ComponentResult Media3DGetRendererList (
   MediaHandler mh,
   QTAtomContainer *rendererList
);
```

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Movies.h`

Media3DGetViewObject

Deprecated.

```
ComponentResult Media3DGetViewObject (
   MediaHandler mh,
   void *tq3viewObject
);
```

Version Notes
Introduced in QuickTime 4.1.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Movies.h`

Media3DRotateNamedObjectTo

Deprecated.

```
ComponentResult Media3DRotateNamedObjectTo (
   MediaHandler mh,
   char *objectName,
   Fixed xDegrees,
   Fixed yDegrees,
   Fixed zDegrees
);
```

Version Notes
Introduced in QuickTime 4.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Movies.h`

Media3DScaleNamedObjectTo

Deprecated.

```
ComponentResult Media3DScaleNamedObjectTo (
   MediaHandler mh,
   char *objectName,
   Fixed xScale,
   Fixed yScale,
   Fixed zScale
);
```

Return Value

Version Notes
Introduced in QuickTime 4.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Movies.h`

Media3DSetCameraAngleAspect

Deprecated.

```
ComponentResult Media3DSetCameraAngleAspect (
   MediaHandler mh,
   QTFloatSingle fov,
   QTFloatSingle aspectRatioXToY
);
```

Version Notes
Introduced in QuickTime 4.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Movies.h`

Media3DSetCameraData

Deprecated.

```
ComponentResult Media3DSetCameraData (
   MediaHandler mh,
   void *cameraData
);
```

Version Notes
Introduced in QuickTime 4.

Availability
Available in Mac OS X v10.0 and later.

Declared In
Movies.h

Media3DSetCameraRange

Deprecated.

```
ComponentResult Media3DSetCameraRange (
   MediaHandler mh,
   void *tQ3CameraRange
);
```

Version Notes
Introduced in QuickTime 4.

Availability
Available in Mac OS X v10.0 and later.

Declared In
Movies.h

Media3DTranslateNamedObjectTo

Deprecated.

```
ComponentResult Media3DTranslateNamedObjectTo (
   MediaHandler mh,
   char *objectName,
   Fixed x,
   Fixed y,
   Fixed z
);
```

Version Notes
Introduced in QuickTime 4.

Availability
Available in Mac OS X v10.0 and later.

Declared In
Movies.h

MovieMediaGetChildDoMCActionCallback

Undocumented

```
ComponentResult MovieMediaGetChildDoMCActionCallback (
   MediaHandler mh,
   DoMCActionUPP *doMCActionCallbackProc,
   long *refcon
);
```

Parameters

mh

A media handler. You can obtain this reference from `GetMediaHandler`.

doMCActionCallbackProc

A pointer to a Universal Procedure Pointer that accesses a `DoMCActionProc` callback.

refcon

A pointer to a reference constant to be passed to your callback. Use this constant to point to a data structure containing any information your callback needs.

Return Value

You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Version Notes

Introduced in QuickTime 4.1.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Movies.h`

MovieMediaGetChildMovieDataReference

Undocumented

```
ComponentResult MovieMediaGetChildMovieDataReference (
   MediaHandler mh,
   QTAtomID dataRefID,
   short dataRefIndex,
   OSType *dataRefType,
   Handle *dataRef,
   QTAtomID *dataRefIDOut,
   short *dataRefIndexOut
);
```

Parameters

mh

A media handler. You can obtain this reference from `GetMediaHandler`.

dataRefID

Undocumented

dataRefIndex

Undocumented

dataRefType

Undocumented

dataRef

Undocumented

dataRefIDOut

Undocumented

dataRefIndexOut

Undocumented

Return Value
You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Version Notes
Introduced in QuickTime 4.1.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Movies.h`

MovieMediaGetCurrentMovieProperty

Retrieves current properties from a media handler's movie.

```
ComponentResult MovieMediaGetCurrentMovieProperty (
   MediaHandler mh,
   OSType whichProperty,
   void *value
);
```

Parameters

mh

A media handler. You can obtain this reference from `GetMediaHandler`.

whichProperty

A constant (see below) that designates the property to be retrieved. See these constants:

`kMoviePropertyDuration`
`kMoviePropertyTimeScale`
`kMoviePropertyTime`
`kMoviePropertyNaturalBounds`
`kMoviePropertyMatrix`
`kMoviePropertyTrackList`

value

A pointer to the returned property value.

Return Value
You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Version Notes
Introduced in QuickTime 4.1.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Movies.h`

MovieMediaGetCurrentTrackProperty

Retrieves the media type property from a media handler's track.

```
ComponentResult MovieMediaGetCurrentTrackProperty (
   MediaHandler mh,
   long trackID,
   OSType whichProperty,
   void *value
);
```

Parameters

mh

A media handler. You can obtain this reference from `GetMediaHandler`.

trackID

The ID value of the track for this operation.

whichProperty

A constant (see below) that designates the property to be retrieved. Only the track's media type property constant is currently defined. See these constants:

`kTrackPropertyMediaType`

value

A pointer to the returned property value.

Return Value
You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Version Notes
Introduced in QuickTime 4.1.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Movies.h`

MovieMediaGetDoMCActionCallback

Gets a DoMCActionProc callback for a media.

```
ComponentResult MovieMediaGetDoMCActionCallback (
   MediaHandler mh,
   DoMCActionUPP *doMCActionCallbackProc,
   long *refcon
);
```

Parameters

mh

A media handler. You can obtain this reference from `GetMediaHandler`.

doMCActionCallbackProc

A pointer to a Universal Procedure Pointer that accesses a `DoMCActionProc` callback.

refcon

A pointer to a reference constant to be passed to your callback. Use this constant to point to a data structure containing any information your callback needs.

Return Value

You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Version Notes

Introduced in QuickTime 4.1.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Movies.h`

MovieMediaLoadChildMovieFromDataReference

Undocumented

```
ComponentResult MovieMediaLoadChildMovieFromDataReference (
   MediaHandler mh,
   QTAtomID dataRefID
);
```

Parameters

mh

A media handler. You can obtain this reference from `GetMediaHandler`.

dataRefID

Undocumented

Return Value

You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Version Notes

Introduced in QuickTime 4.1.

Availability

Available in Mac OS X v10.0 and later.

Declared In
Movies.h

MovieMediaSetChildMovieDataReference

Undocumented

```
ComponentResult MovieMediaSetChildMovieDataReference (
   MediaHandler mh,
   QTAtomID dataRefID,
   OSType dataRefType,
   Handle dataRef
);
```

Parameters

mh
A media handler. You can obtain this reference from GetMediaHandler.

dataRefID
Undocumented

dataRefType
Undocumented

dataRef
Undocumented

Return Value
You can access Movie Toolbox error returns through GetMoviesError (page 70) and GetMoviesStickyError (page 71), as well as in the function result. See Error Codes.

Version Notes
Introduced in QuickTime 4.1.

Availability
Available in Mac OS X v10.0 and later.

Declared In
Movies.h

MoviesTask

Services active movies.

```
void MoviesTask (
   Movie theMovie,
   long maxMilliSecToUse
);
```

Parameters

theMovie
The movie for this operation. Your application obtains this movie identifier from such functions as NewMovie (page 108), NewMovieFromFile, and NewMovieFromHandle. If you set this parameter to NIL, the Movie Toolbox services all of your active movies.

maxMilliSecToUse

Determines the maximum number of milliseconds that MoviesTask can work before returning. If this parameter is 0, MoviesTask services every active movie exactly once and then returns. If the parameter is nonzero, MoviesTask services as many movies as it can in the allotted time before returning. Once the MoviesTask function starts servicing a movie, it cannot stop until it has completely met the requirements of the movie. Consequently, the MoviesTask function may execute for a longer time than that specified in maxMilliSecToUse. However, the function does not start servicing a new movie if the time specified by maxMilliSecToUse has elapsed. The preferred way to use MoviesTask is to set the maxMilliSecToUse parameter to 0; however, if you just want to play one movie, you can call MoviesTask on that one. If your rate is 0, MoviesTask draws that frame and no other.

Return Value

You can access error returns from this function through GetMoviesError (page 70) and GetMoviesStickyError (page 71). See Error Codes.

Discussion

You should call MoviesTask as often as possible from your application's main event loop. Note that you should call this function after you have performed your own event processing. MoviesTask services only active movies, and only enabled tracks within those active movies.

Special Considerations

Note that the MoviesTask function services only your movies. Your application must give other applications the opportunity to call MoviesTask for their movies.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

CompressMovies

Graphic Import-Export

MovieGWorlds

vrscript

vrscript.win

Declared In

Movies.h

NewCallBack

Creates a new callback event.

```
QTCallBack NewCallBack (
   TimeBase tb,
   short cbType
);
```

Parameters

tb

The callback event's time base. You obtain this identifier from NewTimeBase (page 110).

cbType

Constants (see below) that specify when the callback event is to be invoked. The value of this field governs how the Movie Toolbox interprets the data supplied in the param1, param2, and param3 parameters to the `CallMeWhen` (page 27) function. In addition, if the high-order bit of the `cbType` parameter is set to 1 (this bit is defined by the `callBackAtInterrupt` flag), the event can be invoked at interrupt time. See these constants:

```
callBackAtTime
callBackAtRate
callBackAtTimeJump
callBackAtExtremes
callBackAtInterrupt
```

Return Value
A pointer to a `CallBackRecord` structure containing the new callback event.

Special Considerations

The callback event created is not active until you schedule it by calling the `CallMeWhen` (page 27) function. You must not call this function at interrupt time.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
qtbigscreen
qtbigscreen.win
Show Movie
SimpleCocoaMovie
SimpleCocoaMovieQT

Declared In
`Movies.h`

NewMovie

Creates a new movie in memory.

```
Movie NewMovie (
   long flags
);
```

Parameters

flags

Flags (see below) that specify control information for the new movie. Be sure to set unused flags to 0.

Return Value
The identifier for the new movie. If `NewMovie` fails, the returned identifier is set to `NIL`. You can use `GetMoviesError` (page 70) to obtain the error result, or `noErr` if there was no error. See `Error Codes`.

Discussion
You can use `NewMovie` to create a new empty movie, which contains no tracks. The Movie Toolbox initializes the data structures for the new movie. Your application assigns the data to the movie by calling the functions that are described in `NewMovieTrack`.

The Movie Toolbox sets many movie characteristics to default values. If you want to change these defaults, your application must call other Movie Toolbox functions. For example, the Movie Toolbox sets the movie's graphics world to the one that is active when you call `NewMovie`. To change the graphics world for the new movie, your application should use `SetMovieGWorld` (page 139). The default QuickTime movie time scale is 600 units per second; however, this number may change in the future. The default time scale was chosen because it is convenient for working with common video frame rates of 30, 25, 24, 15, 12, 10, and 8.

Special Considerations

The Movie Toolbox automatically sets the movie's graphics world based on the current graphics port. Be sure that your application's graphics port is valid before you call this function, even if the movie is sound-only; you can use `GetGWorld` to check for a valid port, or you can use `NewGWorld` to create a port. The graphics port must remain valid for the life of the movie or until you set another valid graphics port for the movie using `SetMovieGWorld` (page 139).

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
Graphic Import-Export
mfc.win
qtdataref
qtdataref.win
SoundPlayer.win

Declared In
`Movies.h`

NewMovieFromProperties

Creates a new movie using movie properties.

```
OSStatus NewMovieFromProperties (
   ItemCount inputPropertyCount,
   QTNewMoviePropertyElement *inputProperties,
   ItemCount outputPropertyCount,
   QTNewMoviePropertyElement *outputProperties,
   Movie *theMovie
);
```

Parameters

`inputPropertyCount`
The number of properties in the array passed in `inputProperties`.

`inputProperties`
A pointer to a property array describing how to instantiate the movie. See `QTNewMoviePropertyElement`.

outputPropertyCount

The number of properties in the array passed in `outputProperties`.

outputProperties

A pointer to a property array to receive output parameters. See `QTNewMoviePropertyElement`. You may pass NULL if you don't want this information. The caller is responsible for calling the appropriate routines to dispose of any property values returned here. Since callers specify the property classes and IDs, they know who to call to dispose of the property values.

theMovie

A pointer to a variable that receives the new movie.

Return Value
An error code. Returns `memFullErr` if the function could not allocate memory, `paramErr` if `inputProperties` or `theMovie` is NULL, or `noErr` if there is no error.

Discussion
This function can be used in all the cases where an existing `NewMovieFrom...` call is used. When calling this function, you supply a set of input properties that describe the information required to instantiate the movie (its data reference, audio context, visual context, and so on). You can also supply a set of output properties that you may be interested in; for example, information about whether the data reference was changed. See `New Movie Property Codes`.

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

Related Sample Code
MovieVideoChart
QTPixelBufferVCToCGImage
QTSetMovieAudioDevice
SimpleAudioExtraction
SimpleHIMovieViewPlayer

Declared In
`Movies.h`

NewTimeBase

Obtains a new time base.

```
TimeBase NewTimeBase (
   void
);
```

Return Value
The ID of the new time base.

Discussion
This function sets the rate of the time base to 0, the start time to its minimum value, the time value to 0, and the stop time to its maximum value. The function assigns the default clock component to the new time base.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
qteffects.win
QTMusicToo
qtshoweffect
VideoProcessing
vrscript.win

Declared In
`Movies.h`

PlayMoviePreview

Plays a movie's preview.

```
void PlayMoviePreview (
   Movie theMovie,
   MoviePreviewCallOutUPP callOutProc,
   long refcon
);
```

Parameters

`theMovie`

The movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie` (page 108), `NewMovieFromFile`, and `NewMovieFromHandle`.

`callOutProc`

A pointer to a `MoviePreviewCallOutProc` callback in your application. The Movie Toolbox calls this function repeatedly while the movie preview is playing. You can use this function to stop the preview. If you don't want to assign a function, set this parameter to `NIL`.

`refcon`

A reference constant that the Movie Toolbox passes to your callback. Use this parameter to point to a data structure containing any information your callback needs.

Return Value
You can access error returns from this function through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71). See `Error Codes`.

Discussion
This function sets the movie into preview mode, plays the movie preview, sets the movie back to normal playback mode, and returns to your application. The Movie Toolbox plays the preview in the movie's graphics world. Note that if you call the `GetMovieActiveSegment` (page 53) function from within your movie callout function, the Movie Toolbox will have changed the active movie segment to be the preview segment of the movie. The Movie Toolbox restores the active segment when the preview is done playing.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
qtinfo
qtinfo.win

Declared In
`Movies.h`

PrePrerollMovie

Sets up any necessary network connections to receive streaming content.

```
OSErr PrePrerollMovie (
   Movie m,
   TimeValue time,
   Fixed rate,
   MoviePrePrerollCompleteUPP proc,
   void *refcon
);
```

Parameters

m

The movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie` (page 108), `NewMovieFromFile`, and `NewMovieFromHandle`.

time

The starting time of the movie segment to play.

rate

The rate at which you anticipate playing the movie. You specify the movie rate as a 32-bit, fixed-point number. Positive integers indicate forward rates and negative integers indicate reverse rates.

proc

The `MoviePrePrerollCompleteProc` callback you want called when pre-prerolling is complete. If a completion `proc` is specified, `PrePrerollMovie` operates asynchronously. You must call `MoviesTask` periodically during asynchronous operation. If no completion `proc` is specified, `PrePrerollMovie` operates synchronously.

refcon

A reference constant that is passed to your callback. Use this parameter to point to a data structure containing any information your callback needs.

Return Value
You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Discussion
Before a movie is played, it is normally prerolled. During preroll, the Movie Toolbox tells the appropriate media handlers to load the movie data, allocate sound channels, start up image-decompression sequences, and so on. Before a movie that contains streaming content is prerolled, it must be pre-prerolled. This sets up any necessary network connections between the client and the server. If a movie contains streaming content (one or more `'strm'` tracks), you must call this function before calling `PrerollMovie` (page 113). If the movie does not contain streaming content, calling this function has no effect. If your application calls `PrerollMovie`, it should always call this function first. If you play movies using a movie controller, you don't need to preroll or pre-preroll the movie explicitly; it is done for you automatically. If a completion `proc` is specified in the `proc` parameter, this function operates asynchronously; it returns almost immediately and calls the completion `proc` when pre-prerolling is complete.

Special Considerations

You must call `MoviesTask` (page 106) periodically to grant time for pre-prerolling during asynchronous operation. If no completion `proc` is specified, this function operates synchronously; the function will not return until pre-prerolling is complete. This can take a long time, particularly if a dial-up network connection must be established.

Version Notes

Introduced in QuickTime 4. Beginning with QuickTime 4, your application should call this function any time it calls `PrerollMovie` (page 113).

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

qtfullscreen

qtfullscreen.win

vrscript

vrscript.win

Declared In

`Movies.h`

PrerollMovie

Prepares a portion of a movie for playback.

```
OSErr PrerollMovie (
   Movie theMovie,
   TimeValue time,
   Fixed Rate
);
```

Parameters

theMovie

The movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie` (page 108), `NewMovieFromFile`, and `NewMovieFromHandle`.

time

The starting time of the movie segment to play.

Rate

The rate at which you anticipate playing the movie. You specify the movie rate as a 32-bit, fixed-point number. Positive integers indicate forward rates and negative integers indicate reverse rates.

Return Value

You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Discussion

When your application calls `PrerollMovie`, the Movie Toolbox tells the appropriate media handlers to prepare to play the movie. The media handlers may then load the movie data and perform any other necessary preparations to play the movie, such as allocating sound channels and starting up image-decompression sequences. In this manner, you can eliminate playback stutter when the movie starts playing.

If your application uses QuickTime's Movie Toolbox to play back movies, there are two choices for how to preroll the movie. Like the movie controller, the Movie Toolbox provides a single function call, `StartMovie` (page 181), which will both preroll the movie and start it playing. Unlike the movie controller, the Movie Toolbox function doesn't allow you to specific the rate to play the movie at, but instead assumes the movie's preferred rate.

Calling `StartMovie`, just like the movie controller's preroll and play action, first prerolls the movie and then sets it playing. If your application requires more control, the Movie Toolbox provides lower level functions that give you more control:

```
// PrerollMovie coding example
StartMovie(theMovie);

TimeValue timeNow;
Fixed playRate;
timeNow =GetMovieTime(theMovie, NIL);
playRate =GetMoviePreferredRate(theMovie);
PrePrerollMovie(theMovie, timeNow, playRate, NIL, NIL);
PrerollMovie(theMovie, timeNow, playRate);
SetMovieRate(theMovie, playRate);
```

Special Considerations

You should always call `PrePrerollMovie` (page 112) before calling this function.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
DigitizerShell
LiveVideoMixer2
LiveVideoMixer3
MovieBrowser
MovieGWorlds

Declared In
`Movies.h`

PutMovieForDataRefIntoHandle

Puts a self-contained movie into a handle.

```
OSErr PutMovieForDataRefIntoHandle (
   Movie theMovie,
   Handle dataRef,
   OSType dataRefType,
   Handle publicMovie
);
```

Parameters

theMovie

The movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie` (page 108), `NewMovieFromFile`, and `NewMovieFromHandle`.

dataRef

A handle to the storage in which the movie will be written.

dataRefType

The data reference type. See `Data References`.

publicMovie

The handle that is to receive the new movie resource. The function resizes the handle if necessary.

Return Value

You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Discussion

If the data reference and data reference type is passed, all media references to the same storage are converted to self-references in the resulting public movie handle. This `'moov'` atom can be then written to the storage.

Version Notes

Introduced in QuickTime 6.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`Movies.h`

PutMovieIntoDataFork

Stores a movie in the data fork of a given file.

```
OSErr PutMovieIntoDataFork (
   Movie theMovie,
   short fRefNum,
   long offset,
   long maxSize
);
```

Parameters

theMovie

The movie to be stored in the data fork of an atom. Your application obtains this movie identifier from such functions as `NewMovie` (page 108), `NewMovieFromFile`, and `NewMovieFromHandle`.

fRefNum

A file reference number for the data fork of the given file. You pass in an open write path in the `fRefNum` parameter.

offset
Indicates where the movie should be written.

maxSize
The largest number of bytes that may be written.

Return Value
You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Movies.h`

PutMovieIntoDataFork64

Provides a 64-bit version of PutMovieIntoDataFork.

```
OSErr PutMovieIntoDataFork64 (
   Movie theMovie,
   long fRefNum,
   const wide *offset,
   unsigned long maxSize
);
```

Parameters

theMovie
A movie identifier. Your application obtains this identifier from such functions as `NewMovie` (page 108), `NewMovieFromFile`, and `NewMovieFromHandle`.

fRefNum
A file reference number for the data fork of the given file. You pass in an open write path in the `fRefNum` parameter.

offset
Pointer to a 64-bit value that indicates where the movie should be written.

maxSize
The largest number of bytes that may be written.

Return Value
You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Version Notes
Introduced in QuickTime 4. Superseded in QuickTime 6 by `PutMovieIntoStorage` (page 117).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Movies.h`

PutMovieIntoHandle

Creates a new movie resource.

```
OSErr PutMovieIntoHandle (
   Movie theMovie,
   Handle publicMovie
);
```

Parameters

theMovie

The movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie` (page 108), `NewMovieFromFile`, and `NewMovieFromHandle`.

publicMovie

The handle that is to receive the new movie resource. The function resizes the handle if necessary.

Return Value

You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Discussion

Use this handle to store a QuickTime movie in a specialized storage format.

Special Considerations

Note that you cannot use this new movie with other Movie Toolbox functions, except for `NewMovieFromHandle`.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

CompressMovies

DragAndDrop Shell

QTAudioExtractionPanel

QTExtractAndConvertToAIFF

ThreadsExportMovie

Declared In

`Movies.h`

PutMovieIntoStorage

Writes a movie to a storage location managed by a data handler.

```
OSErr PutMovieIntoStorage (
   Movie theMovie,
   DataHandler dh,
   const wide *offset,
   unsigned long maxSize
);
```

Parameters

theMovie

The movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie` (page 108), `NewMovieFromFile`, and `NewMovieFromHandle`.

dh

A data handler for the data fork of the storage container. You pass an open write path in this parameter.

offset

A pointer to a value that indicates where the movie should be written in the container.

maxSize

The largest number of bytes that may be written.

Return Value

You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Discussion

If you are writing a custom data handler, make sure it supports `DataHGetDataRef`. It must also support `DataHWrite64`, or `DataHWrite` if 64-bit offsets are not supported.

Version Notes

Introduced in QuickTime 6. This function supersedes `PutMovieIntoDataFork64` (page 116).

Availability

Available in Mac OS X v10.2 and later.

Declared In

`Movies.h`

QTAudioContextCreateForAudioDevice

Creates a QTAudioContext object that encapsulates a connection to a CoreAudio output device.

```
OSStatus QTAudioContextCreateForAudioDevice (
   CFAllocatorRef allocator,
   CFStringRef audioDeviceUID,
   CFDictionaryRef options,
   QTAudioContextRef *newAudioContextOut
);
```

Parameters

allocator

Allocator used to create the audio context.

coreAudioDeviceUID

CoreAudio device UID. NULL means the default device.

options

Reserved. Pass NULL.

newAudioContextOut

Points to a variable to receive the new audio context.

Return Value
An error code. Returns `noErr` if there is no error.

Discussion
This routine creates a `QTAudioContext` object that encapsulates a connection to a CoreAudio output device. This object is suitable for passing to `SetMovieAudioContext` or `NewMovieFromProperties` (page 109), which targets the audio output of the movie to that device. A `QTAudioContext` object cannot be associated with more than one movie. Each movie needs its own connection to the device. In order to play more than one movie to a particular device, create a `QTAudioContext` object for each movie. You are responsible for releasing the `QTAudioContext` object created by this routine. After calling `SetMovieAudioContext` or `NewMovieFromProperties` (page 109), you can release the object since these APIs will retain it for their own use.

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

Related Sample Code
QTSetMovieAudioDevice

Declared In
`Movies.h`

QTGetTimeUntilNextTask

Reports the duration until the next time QuickTime needs to run a task.

```
OSErr QTGetTimeUntilNextTask (
   long *duration,
   long scale
);
```

Parameters

duration

A pointer to the duration until the next time QuickTime needs access to the processor. If the returned duration is 0, QuickTime needs to run a task immediately.

scale

The time scale in which to express the returned duration. For example, pass 60 if you want the duration value expressed in ticks (60ths of a second).

Return Value
You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Discussion
Periodically, applications have to give processing time to QuickTime by calling a function such as `MCIsPlayerEvent`. Instead of routinely calling `MCIsPlayerEvent` 10 to 20 times per second, you can call `QTGetTimeUntilNextTask` to determine when QuickTime next needs access to the processor. The result is a more efficent use of processor resources. To handle cases when QuickTime may need to run a task earlier than projected by this function, you can install a `QTNextTaskNeededSoonerCallbackProc` callback.

Version Notes
Introduced in QuickTime 6.

Availability
Available in Mac OS X v10.2 and later.

Related Sample Code
QTCarbonCoreImage101
qtshellCEvents
qtshellCEvents.win
VideoProcessing

Declared In
Movies.h

QTGetWallClockTimeBase

Returns the system's real-time time base.

```
OSErr QTGetWallClockTimeBase (
   TimeBase *wallClockTimeBase
);
```

Parameters

wallClockTimeBase
A pointer to the wall clock's time base.

Return Value
You can access Movie Toolbox error returns through GetMoviesError (page 70) and GetMoviesStickyError (page 71), as well as in the function result. See Error Codes.

Version Notes
Introduced in QuickTime 6.

Availability
Available in Mac OS X v10.2 and later.

Declared In
Movies.h

QTIdleManagerClose

Closes the Mac OS Idle Manager.

```
OSErr QTIdleManagerClose (
   IdleManager im
);
```

Parameters

im
A pointer to the opaque data structure that was returned by QTIdleManagerOpen (page 122).

Return Value
You can access Movie Toolbox error returns through GetMoviesError (page 70) and GetMoviesStickyError (page 71), as well as in the function result. See Error Codes.

Discussion
Your application should call this function after it no longer needs access to the Idle Manager.

Version Notes
Introduced in QuickTime 6.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`Movies.h`

QTIdleManagerGetNextIdleTime

Retrieves the next idle time known to the Idle Manager.

```
OSErr QTIdleManagerGetNextIdleTime (
   IdleManager im,
   TimeRecord *nextIdle
);
```

Parameters

im

A pointer to an opaque data structure that belongs to the Mac OS Idle Manager. You get this pointer by calling `QTIdleManagerOpen` (page 122).

nextIdle

A pointer to the next idle time.

Return Value
You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Version Notes
Introduced in QuickTime 6.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`Movies.h`

QTIdleManagerNeedsAnIdle

Tells the Idle Manager whether an idle will be required.

```
OSErr QTIdleManagerNeedsAnIdle (
   IdleManager im,
   Boolean *needsOne
);
```

Parameters

im

A pointer to an opaque data structure that belongs to the Mac OS Idle Manager. You get this pointer by calling `QTIdleManagerOpen` (page 122).

needsOne

Pass a pointer to a variable; on return, TRUE means that an idle will be required, FALSE means no idle will be required.

Return Value
You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Version Notes
Introduced in QuickTime 6.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`Movies.h`

QTIdleManagerOpen

Opens the Mac OS Idle Manager.

```
IdleManager QTIdleManagerOpen (
   void
);
```

Return Value
A pointer to an opaque data structure that belongs to the Mac OS Idle Manager.

Discussion
You must call this function before using the Mac OS Idle Manager.

Version Notes
Introduced in QuickTime 6.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`Movies.h`

QTIdleManagerSetNextIdleTime

Informs the idle manager of the next required idle time.

```
OSErr QTIdleManagerSetNextIdleTime (
   IdleManager im,
   TimeRecord *nextIdle
);
```

Parameters

im

A pointer to an opaque data structure that belongs to the Mac OS Idle Manager. You get this pointer by calling `QTIdleManagerOpen` (page 122).

nextIdle

A pointer to the time of the next required idle.

Return Value
You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Discussion
If a media handler needs to call this function, you must do wallclock time calculations. That means you may need to call `QTGetWallClockTimeBase` (page 120) and `ConvertTime` (page 31) to convert from track time or media time to wallclock time, plus `ConvertTimeScale` (page 31) to convert to the timescale you like to work in.

Version Notes
Introduced in QuickTime 6.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`Movies.h`

QTIdleManagerSetNextIdleTimeDelta

Informs the idle manager of the time from the currently set idle time to the next idle time required after it.

```
OSErr QTIdleManagerSetNextIdleTimeDelta (
   IdleManager im,
   TimeValue duration,
   TimeScale scale
);
```

Parameters

im

A pointer to an opaque data structure that belongs to the Mac OS Idle Manager. You get this pointer by calling `QTIdleManagerOpen` (page 122).

duration

The time from the current idle time to the next one.

scale

The time scale in which the duration is expressed.

Return Value
You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Discussion
This routine lets you pass in a duration and a scale and it gives you a single idle. For example, if you need an idle a half second from now, you can pass in a duration of 500 and a scale of 1000, or a duration of 1 and scale of 2. This will get you one idle 0.5 seconds from now.

Special Considerations

Every time you get idled, you need to call this function again to set your next idle. If you don't, QuickTime will assume a default duration to the next idle of 0 and you'll be idled all the time.

Version Notes
Introduced in QuickTime 6.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`Movies.h`

QTIdleManagerSetNextIdleTimeNever

Sets the next idle time indefinitely in the future.

```
OSErr QTIdleManagerSetNextIdleTimeNever (
   IdleManager im
);
```

Parameters

im

A pointer to an opaque data structure that belongs to the Mac OS Idle Manager. You get this pointer by calling `QTIdleManagerOpen` (page 122).

Return Value
You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Version Notes
Introduced in QuickTime 6.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`Movies.h`

QTIdleManagerSetNextIdleTimeNow

Requests an idle as soon as possible.

```
OSErr QTIdleManagerSetNextIdleTimeNow (
   IdleManager im
);
```

Parameters

im

A pointer to an opaque data structure that belongs to the Mac OS Idle Manager. You get this pointer by calling `QTIdleManagerOpen` (page 122).

Return Value
You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Version Notes
Introduced in QuickTime 6.

Availability
Available in Mac OS X v10.2 and later.

Declared In
Movies.h

QTIdleManagerSetParent

Sets the parent of an Idle Manager instance.

```
OSErr QTIdleManagerSetParent (
   IdleManager im,
   IdleManager parent
);
```

Parameters

im

A pointer to an opaque data structure that belongs to the Mac OS Idle Manager. You get this pointer by calling QTIdleManagerOpen (page 122).

parent

A pointer to a different Idle Manager data structure. You get this pointer also by calling QTIdleManagerOpen (page 122).

Return Value
You can access Movie Toolbox error returns through GetMoviesError (page 70) and GetMoviesStickyError (page 71), as well as in the function result. See Error Codes.

Version Notes
Introduced in QuickTime 6.

Availability
Available in Mac OS X v10.2 and later.

Declared In
Movies.h

QTInstallNextTaskNeededSoonerCallback

Installs a QTNextTaskNeededSoonerCallbackProc callback.

```
OSErr QTInstallNextTaskNeededSoonerCallback (
   QTNextTaskNeededSoonerCallbackUPP callbackProc,
   TimeScale scale,
   unsigned long flags,
   void *refcon
);
```

Parameters

callbackProc

A Universal Procedure Pointer to a QTNextTaskNeededSoonerCallbackProc callback.

scale

The time scale that QuickTime will use when reporting the duration until the next time QuickTime needs to be called, via QTGetTimeUntilNextTask (page 119).

flags

Unused; set to 0.

refcon

A reference constant to be passed to your callback. Use this parameter to point to a data structure containing any information your callback needs.

Return Value

You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Discussion

This routine installs a callback procedure that specifies when QuickTime next needs to be tasked. The callback procedure may be called at interrupt time or from another Mac OS X thread, so you must be careful not to cause race conditions. You can install or uninstall multiple callback procedures if necessary; they will be called in sequence. You can also install the same callback multiple times with different `refcon` values, in which case it will be called once with each `refcon` value.

Version Notes

Introduced in QuickTime 6.

Availability

Available in Mac OS X v10.2 and later.

Related Sample Code

QTCarbonCoreImage101

qtshellCEvents

qtshellCEvents.win

VideoProcessing

Declared In

`Movies.h`

QTParseTextHREF

Undocumented

```
OSErr QTParseTextHREF (
   char *href,
   SInt32 hrefLen,
   QTAtomContainer inContainer,
   QTAtomContainer *outContainer
);
```

Parameters

href

A pointer to an HREF string.

hrefLen

The length of the HREF string.

inContainer

Undocumented

outContainer

Undocumented

Return Value
You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Version Notes
Introduced in QuickTime 4.1.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Movies.h`

QTSoundDescriptionConvert

Converts a sound description from one version to another.

```
OSStatus QTSoundDescriptionConvert (
   QTSoundDescriptionKind fromKind,
   SoundDescriptionHandle fromDescription,
   QTSoundDescriptionKind toKind,
   SoundDescriptionHandle *toDescription
);
```

Parameters

`fromKind`
Reserved. Set to `kSoundDescriptionKind_Movie_AnyVersion`.

`fromDescription`
A handle to the sound description to be converted.

`toKind`
The version you want `fromDescription` to be.

`toDescription`
A reference to the resulting `SoundDescription` structure. You must dispose of the reference using `DisposeHandle`.

Return Value
An error code. Returns `noErr` if there is no error.

Discussion
The `fromKind` parameter is reserved for future expansion; at present you must set it to `kQTSoundDescriptionKind_Movie_AnyVersion`. Depending on the value you pass in `toKind`, you can specify that you would like a specific `SoundDescription` version, the lowest possible version (given the constraints of the format described by `fromDescription`), or any version at all. Use these constants:

```
enum {
  kQTSoundDescriptionKind_Movie_Version1 = 'mvv1',
  kQTSoundDescriptionKind_Movie_Version2 = 'mvv2',
  kQTSoundDescriptionKind_Movie_LowestPossibleVersion = 'mvlo',
  kQTSoundDescriptionKind_Movie_AnyVersion = 'mvny'
};
```

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

Declared In
`Movies.h`

QTSoundDescriptionCreate

Creates a sound description structure of the requested kind from an AudioStreamBasicDescription, optional audio channel layout, and optional magic cookie.

```
OSStatus QTSoundDescriptionCreate (
   AudioStreamBasicDescription *inASBD,
   AudioChannelLayout *inLayout,
   ByteCount inLayoutSize,
   void *inMagicCookie,
   ByteCount inMagicCookieSize,
   QTSoundDescriptionKind inRequestedKind,
   SoundDescriptionHandle *outSoundDesc
);
```

Parameters

inASBD
A description of the format.

inLayout
The audio channel layout (can be NULL if there isn't one).

inLayoutSize
The size of the audio channel layout (should be 0 if `inLayout` is NULL).

inMagicCookie
The magic cookie for the decompressor (can be NULL if the decompressor doesn't require one).

inMagicCookieSize
The size of the magic cookie (should be 0 if the `inMagicCookie` parameter is NULL).

inRequestedKind
The kind of sound description to create (see Discussion, below).

outSoundDesc
The resulting sound description. The caller must dispose of it with `DisposeHandle`.

Return Value
An error code. Returns `noErr` if there is no error.

Discussion
The value of `inRequestedKind` can be taken from these values:

```
enum {
  kQTSoundDescriptionKind_Movie_Version1 = 'mvv1',
  kQTSoundDescriptionKind_Movie_Version2 = 'mvv2',
  kQTSoundDescriptionKind_Movie_LowestPossibleVersion = 'mvlo',
  kQTSoundDescriptionKind_Movie_AnyVersion = 'mvny'
};
```

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

Related Sample Code
QTExtractAndConvertToMovieFile

SCAudioCompress
WhackedTV

Declared In
`Movies.h`

QTSoundDescriptionGetProperty

Gets a particular property of a sound description.

```
OSStatus QTSoundDescriptionGetProperty (
   SoundDescriptionHandle inDesc,
   QTPropertyClass inPropClass,
   QTPropertyID inPropID,
   ByteCount inPropValueSize,
   QTPropertyValuePtr outPropValueAddress,
   ByteCount *outPropValueSizeUsed
);
```

Parameters

inDesc
The sound description being interrogated.

inPropClass
The class of the property being requested.

inPropID
The ID of the property being requested.

inPropValueSize
The size of the property value buffer.

outPropValueAddress
A pointer to the property value buffer.

outPropValueSizeUsed
The actual size of the returned property value (can be NULL).

Return Value
An error code. Returns `noErr` if there is no error.

Discussion
The following constants identify sound description properties.

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

Related Sample Code
CaptureAndCompressIPBMovie
WhackedTV

Declared In
`Movies.h`

QTSoundDescriptionGetPropertyInfo

Gets information about a particular property of a sound description.

```
OSStatus QTSoundDescriptionGetPropertyInfo (
   SoundDescriptionHandle inDesc,
   QTPropertyClass inPropClass,
   QTPropertyID inPropID,
   QTPropertyValueType *outPropType,
   ByteCount *outPropValueSize,
   UInt32 *outPropertyFlags
);
```

Parameters

inDesc

The sound description being interrogated.

inPropClass

The class of the property being requested.

inPropID

The ID of the property being requested.

outPropType

The type of the property returned here (can be NULL).

outPropValueSize

The size of the property returned here (can be NULL).

outPropertyFlags

The property flags returned here (can be NULL).

Return Value

An error code. Returns `noErr` if there is no error.

Discussion

The following constants identify sound description properties.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`Movies.h`

QTSoundDescriptionSetProperty

Sets a particular property of a sound description.

```
OSStatus QTSoundDescriptionSetProperty (
   SoundDescriptionHandle inDesc,
   QTPropertyClass inPropClass,
   QTPropertyID inPropID,
   ByteCount inPropValueSize,
   ConstQTPropertyValuePtr inPropValueAddress
);
```

Parameters

inDesc

The sound description being modified.

inPropClass

The class of the property being set.

inPropID

The ID of the property being set.

inPropValueSize

The size of the property value buffer.

inPropValueAddress

A pointer to the property value buffer.

Return Value
An error code. Returns `noErr` if there is no error.

Discussion
The following constants identify sound description properties.

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

Related Sample Code
SCAudioCompress

Declared In
`Movies.h`

QTTextToNativeText

Undocumented

```
OSErr QTTextToNativeText (
   Handle theText,
   long encoding,
   long flags
);
```

Parameters

theText

Undocumented

encoding

Undocumented

flags

Flags (see below) that define the `text` atom type. See these constants:

`kITextAtomType`

`kITextStringAtomType`

Return Value
You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Movies.h`

QTUninstallNextTaskNeededSoonerCallback

Removes a QTNextTaskNeededSoonerCallbackProc callback.

```
OSErr QTUninstallNextTaskNeededSoonerCallback (
   QTNextTaskNeededSoonerCallbackUPP callbackProc,
   void *refcon
);
```

Parameters

callbackProc

A Universal Procedure Pointer to a `QTNextTaskNeededSoonerCallbackProc` callback that you installed by a previous call to `QTInstallNextTaskNeededSoonerCallback` (page 125).

refcon

A pointer to the reference constant that you passed when the callback was installed.

Return Value
You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Discussion
You pass this routine both a pointer to a callback procedure and a pointer to its reference constant, so you can uninstall one instance of a callback that you installed more than once with different `refcon` values.

Version Notes
Introduced in QuickTime 6.

Availability
Available in Mac OS X v10.2 and later.

Related Sample Code
QTCarbonCoreImage101

VideoProcessing

Declared In
`Movies.h`

RemoveCallBackFromTimeBase

Removes a callback event from the list of scheduled callback events.

```
OSErr RemoveCallBackFromTimeBase (
   QTCallBack cb
);
```

Parameters

cb

The callback event for the operation. Your clock component obtains this value from the parameters passed to your `ClockCallMeWhen` function.

Return Value

You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Discussion

Your clock component should call this function when your `ClockCancelCallBack` function determines that your component can cancel the callback event.

Special Considerations

Your component should call this function only for callback events that were successfully added to the schedule with `AddCallBackToTimeBase` (page 26).

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Movies.h`

SetMovieActive

Activates or deactivates a movie.

```
void SetMovieActive (
   Movie theMovie,
   Boolean active
);
```

Parameters

theMovie

The movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie` (page 108), `NewMovieFromFile`, and `NewMovieFromHandle`.

active

Activates or deactivates the `movie`. Set this parameter to TRUE to activate the `movie`; set this parameter to FALSE to deactivate the movie.

Return Value

You can access error returns from this function through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71). See `Error Codes`.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
BurntTextSampleCode
QTCarbonShell
qtcontroller
qtshellCEvents.win
vrcursors

Declared In
`Movies.h`

SetMovieActiveSegment

Defines a movie's active segment.

```
void SetMovieActiveSegment (
   Movie theMovie,
   TimeValue startTime,
   TimeValue duration
);
```

Parameters

theMovie
The movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie` (page 108), `NewMovieFromFile`, and `NewMovieFromHandle`.

startTime
A time value specifying the starting point of the active segment. Set this parameter to -1 to make the entire movie active. In this case, the `SetMovieActiveSegment` function ignores the `duration` parameter.

duration
A time value that specifies the duration of the active segment. If you are making the entire movie active (by setting the `startTime` parameter to -1), the Movie Toolbox ignores this parameter.

Return Value
You can access error returns from this function through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71). See `Error Codes`.

Discussion
Your application defines the active segment by specifying the starting time and duration of the segment. These values must be expressed in the movie's time coordinate system. By default, the entire movie is active.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
Movies.h

SetMovieAudioContext

Targets a movie to render into an audio context.

```
OSStatus SetMovieAudioContext (
   Movie movie,
   QTAudioContextRef audioContext
);
```

Parameters

movie

The movie.

audioContext

The audio context that the movie will render into.

Return Value
An error code. Returns noErr if there is no error. .

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

Related Sample Code
QTSetMovieAudioDevice

Declared In
Movies.h

SetMovieBox

Sets a movie's boundary rectangle.

```
void SetMovieBox (
   Movie theMovie,
   const Rect *boxRect
);
```

Parameters

theMovie

The movie for this operation. Your application obtains this movie identifier from such functions as NewMovie (page 108), NewMovieFromFile, and NewMovieFromHandle.

boxRect

A pointer to a rectangle that contains the coordinates of the new boundary rectangle.

Return Value
You can access error returns from this function through GetMoviesError (page 70) and GetMoviesStickyError (page 71). See Error Codes.

Discussion
The Movie Toolbox changes the rectangle by modifying the translation and scale values of the movie's matrix to accommodate the new boundary rectangle.

The movie box might not have its upper-left corner set at (0,0) in its display window when the movie is first loaded. Consequently, your application may need to adjust the position of the movie box so that it appears in the appropriate location within your application's document window. If you don't reset the movie position, the movie might not be visible when it starts playing. The following sample code demonstrates how to do this:

```
//Zeroing the boundary rectangle with SetMovieBox
GetMovieBox (movie, &movieBox);
OffsetRect (&movieBox, -movieBox.left, -movieBox.top);
SetMovieBox (movie, &movieBox);
```

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
MovieGWorlds
vrmovies
vrmovies.win
vrscript
vrscript.win

Declared In
`Movies.h`

SetMovieClipRgn

Establishes a movie's clipping region.

```
void SetMovieClipRgn (
   Movie theMovie,
   RgnHandle theClip
);
```

Parameters

`theMovie`
The movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie` (page 108), `NewMovieFromFile`, and `NewMovieFromHandle`.

`theClip`
A handle to the movie's clipping region. The Movie Toolbox makes a copy of this region. Your application must dispose of the region referred to by this parameter when you are done with it. Set this parameter to `NIL` to disable clipping for the movie.

Return Value
You can access error returns from this function through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71). See `Error Codes`.

Discussion
The clipping region is saved with the movie.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
CompressMovies
ConvertToMovieJr
qtcompress
qtcompress.win
VideoProcessing

Declared In
`Movies.h`

SetMovieDisplayClipRgn

Establishes a movie's current display clipping region.

```
void SetMovieDisplayClipRgn (
   Movie theMovie,
   RgnHandle theClip
);
```

Parameters

`theMovie`

The movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie` (page 108), `NewMovieFromFile`, and `NewMovieFromHandle`.

`theClip`

A handle to the movie's display clipping region as a `MacRegion` structure. The Movie Toolbox makes a copy of this region. Your application must dispose of the region referred to by this parameter when you are done with it. Set this parameter to `NIL` to disable a movie's clipping region.

Return Value
You can access error returns from this function through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71). See `Error Codes`.

Discussion
The display clipping region is not saved with the movie. You can access error returns from this function through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71). See `Error Codes`.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Movies.h`

SetMovieDrawingCompleteProc

Assigns a drawing-complete function to a movie.

```
void SetMovieDrawingCompleteProc (
   Movie theMovie,
   long flags,
   MovieDrawingCompleteUPP proc,
   long refCon
);
```

Parameters

theMovie

The movie for this operation.

flags

Contains flags (see below) that control when your drawing complete function is called. See these constants:

`movieDrawingCallWhenChanged`

`movieDrawingCallAlways`

proc

A pointer to your `MovieDrawingCompleteProc` callback. Set this parameter to `NIL` if you want to remove your callback.

refCon

The reference constant you supplied when your application called your callback. Use this parameter to point to a data structure containing any information your callback needs.

Return Value

You can access error returns from this function through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71). See `Error Codes`.

Discussion

The Movie Toolbox calls this function based upon guidelines you establish when you assign the function to the movie.

Special Considerations

Some media handlers may take less efficient playback paths when a drawing-complete function is used, so it should be used only when absolutely necessary.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

ASCIIMoviePlayerSample

bMoviePaletteCocoa

MovieGWorlds

OpenGLMovieQT

VideoProcessing

Declared In

`Movies.h`

SetMovieGWorld

Establishes a movie's display coordinate system by setting the graphics world for displaying the movie.

```
void SetMovieGWorld (
   Movie theMovie,
   CGrafPtr port,
   GDHandle gdh
);
```

Parameters

theMovie

The movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie` (page 108), `NewMovieFromFile`, and `NewMovieFromHandle`.

port

Points to the movie's `CGrafPort` structure or graphics world. Set this parameter to `NIL` to use the current graphics port.

gdh

A handle to the movie's `GDevice` structure. Set this parameter to `NIL` to use the current device. If the `port` parameter specifies a graphics world, set this parameter to `NIL` to use that graphics world's graphics device.

Return Value

You can access error returns from this function through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71). See `Error Codes`.

Special Considerations

When you use this function, the Movie Toolbox remembers the current background color and background pattern. These are used for erasing in the default movie uncover function; see `SetMovieCoverProcs`.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

MovieGWorlds

QTCarbonShell

vrmovies

vrscript

vrscript.win

Declared In

`Movies.h`

SetMovieMasterClock

Assigns a clock component to a movie.

```
void SetMovieMasterClock (
   Movie theMovie,
   Component clockMeister,
   const TimeRecord *slaveZero
);
```

Parameters

theMovie

The movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie` (page 108), `NewMovieFromFile`, and `NewMovieFromHandle`.

clockMeister

The clock component to be assigned to this movie. Your application can obtain this component identifier from `FindNextComponent`.

slaveZero

A pointer to the time, in the clock's time scale, that corresponds to a 0 time value for the `movie`. This parameter allows you to set an offset between the clock component and the time base of the `movie`. Set this parameter to `NIL` if there is no offset.

Return Value

You can access error returns from this function through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71). See `Error Codes`.

Discussion

Don't use `SetTimeBaseMasterClock` (page 153) to assign a clock component to a movie.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Movies.h`

SetMovieMasterTimeBase

Assigns a master time base to a movie.

```
void SetMovieMasterTimeBase (
   Movie theMovie,
   TimeBase tb,
   const TimeRecord *slaveZero
);
```

Parameters

theMovie

The movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie` (page 108), `NewMovieFromFile`, and `NewMovieFromHandle`.

tb

The master time base to be assigned to this movie. Your application obtains this time base identifier from `NewTimeBase` (page 110).

slaveZero

A pointer to the time, in the time scale of the master time base, that corresponds to a 0 time value for the `movie`. This parameter allows you to set an offset between the movie and the master time base. Set this parameter to `NIL` if there is no offset.

Return Value
You can access error returns from this function through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71). See `Error Codes`.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
Show Movie

Declared In
`Movies.h`

SetMovieMatrix

Sets a movie's transformation matrix.

```
void SetMovieMatrix (
   Movie theMovie,
   const MatrixRecord *matrix
);
```

Parameters

theMovie

The movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie` (page 108), `NewMovieFromFile`, and `NewMovieFromHandle`.

matrix

A pointer to the `MatrixRecord` structure for the movie. If you set this parameter to `NIL`, the Movie Toolbox uses the identity matrix.

Return Value
You can access error returns from this function through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71). See `Error Codes`.

Discussion
The Movie Toolbox uses a movie's matrix to map a movie from its display coordinate system to its graphics world.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
OpenGLMovieQT
QTCarbonShell

vrmovies
vrscript
vrscript.win

Declared In
Movies.h

SetMoviePosterTime

Sets the poster time for the movie.

```
void SetMoviePosterTime (
   Movie theMovie,
   TimeValue posterTime
);
```

Parameters

theMovie

The movie for this operation. Your application obtains this movie identifier from such functions as NewMovie (page 108), NewMovieFromFile, and NewMovieFromHandle.

posterTime

The starting time for the movie frame that contains the poster image, expressed in the movie's time coordinate system.

Return Value
You can access error returns from this function through GetMoviesError (page 70) and GetMoviesStickyError (page 71). See Error Codes.

Discussion
Since a movie poster is a still frame, it is defined by a point in time within the movie.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
bMoviePalette
bMoviePaletteCocoa
qtinfo
qtinfo.win
vrmakeobject

Declared In
Movies.h

SetMoviePreferredRate

Specifies a movie's default playback rate.

```
void SetMoviePreferredRate (
   Movie theMovie,
   Fixed rate
);
```

Parameters

theMovie

The movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie` (page 108), `NewMovieFromFile`, and `NewMovieFromHandle`.

rate

The new movie rate as a 32-bit, fixed-point number. Positive integers indicate forward rates and negative integers indicate reverse rates.

Return Value

You can access error returns from this function through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71). See `Error Codes`.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Movies.h`

SetMoviePreferredVolume

Sets a movie's preferred volume setting.

```
void SetMoviePreferredVolume (
   Movie theMovie,
   short volume
);
```

Parameters

theMovie

The movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie` (page 108), `NewMovieFromFile`, and `NewMovieFromHandle`.

volume

The preferred volume setting of the movie. The volume parameter must contain a 16-bit, fixed-point number that contains the movie's default volume. The high-order 8 bits contain the integer part of the value; the low-order 8 bits contain the fractional part. Volume values range from -1.0 to 1.0. Negative values play no sound but preserve the absolute value of the volume setting. You may find the constants shown below useful. See these constants:

Return Value

You can access error returns from this function through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71). See `Error Codes`.

Discussion

A movie's tracks may have their own volume settings. Use `SetTrackVolume` to set the volume of an individual track. A track's volume is scaled by the movie's volume to produce the track's final volume.

Special Considerations

After calling this function you must save the changes it has made, for example by updating or flattening the movie.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
MakeEffectMovie
qteffects.win
qtgraphics.win
QTMusicToo
vrbackbuffer.win

Declared In
`Movies.h`

SetMoviePreviewMode

Places a movie into and out of preview mode.

```
void SetMoviePreviewMode (
   Movie theMovie,
   Boolean usePreview
);
```

Parameters

theMovie

The movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie` (page 108), `NewMovieFromFile`, and `NewMovieFromHandle`.

usePreview

The movie's mode. Set this parameter to TRUE to place the movie into preview mode. Set this parameter to FALSE to place the movie into normal playback mode.

Return Value
You can access error returns from this function through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71). See `Error Codes`.

Discussion
When a movie is in preview mode, only those tracks identified as preview tracks are serviced. You specify how a track is used by calling `SetTrackUsage`.

When you place a movie into preview mode, the Movie Toolbox sets the active movie segment to be the preview segment of the movie. When you take a movie out of preview mode and place it back in normal playback mode, the toolbox sets the active movie segment to be the entire movie. For information about working with active movie segments, see `PrerollMovie` (page 113).

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Movies.h`

SetMoviePreviewTime

Defines the starting time and duration of the movie's preview.

```
void SetMoviePreviewTime (
   Movie theMovie,
   TimeValue previewTime,
   TimeValue previewDuration
);
```

Parameters

theMovie

The movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie` (page 108), `NewMovieFromFile`, and `NewMovieFromHandle`.

previewTime

A time value that specifies the preview's starting time.

previewDuration

A time value that specifies the preview's duration.

Return Value
You can access error returns from this function through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71). See `Error Codes`.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
qtinfo
qtinfo.win
vrmakeobject
vrmakeobject.win

Declared In
`Movies.h`

SetMovieRate

Sets a movie's playback rate.

```
void SetMovieRate (
   Movie theMovie,
   Fixed rate
);
```

Parameters

theMovie

The movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie` (page 108), `NewMovieFromFile`, and `NewMovieFromHandle`.

rate

The new movie rate as a 32-bit, fixed-point number. Positive integers indicate forward rates and negative integers indicate reverse rates. This value immediately changes the rate at which the movie is playing. A value of 1 starts the movie playing at normal speed, a value of 2 causes the movie to play at double speed, -2 starts the movie playing backward at double speed, and so on. A value of 0 stops the movie.

Return Value

You can access error returns from this function through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71). See `Error Codes`.

Discussion

Use this function to change the speed at which a movie is playing. You do not normally use this function to start and stop movies; use the higher level functions `StartMovie` (page 181) and `StopMovie` (page 182) instead. If you start a movie using this function, you should call `PrePrerollMovie` (page 112) and `PrerollMovie` (page 113) first, to set up any network connections, buffers, and data structures necessary to play the movie.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

AddFrameToMovie

OpenGLMovieQT

QTCarbonCoreImage101

Show Movie

vrscript.win

Declared In

`Movies.h`

SetMovieSelection

Sets a movie's current selection.

```
void SetMovieSelection (
   Movie theMovie,
   TimeValue selectionTime,
   TimeValue selectionDuration
);
```

Parameters

theMovie

The movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie` (page 108), `NewMovieFromFile`, and `NewMovieFromHandle`.

selectionTime

A time value specifying the starting point of the current selection.

selectionDuration

A time value that specifies the duration of the current selection.

Return Value

You can access error returns from this function through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71). See `Error Codes`.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

qtinfo

qtinfo.win

SlideShowImporter

SlideShowImporter.win

Declared In

`Movies.h`

SetMoviesErrorProc

Performs custom error notification.

```
void SetMoviesErrorProc (
   MoviesErrorUPP errProc,
   long refcon
);
```

Parameters

errProc

A `MoviesErrorProc` callback.

refcon

A reference constant value. The Movie Toolbox passes this reference constant to your `MoviesErrorProc` callback each time it calls it. Use this parameter to point to a data structure containing any information your callback needs.

Return Value
You can access error returns from this function through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71). See `Error Codes`.

Discussion
Your application must identify its custom error-notification function to the Movie Toolbox. Once you have identified an error-notification function, the Movie Toolbox calls your function each time the current error value is to be set to a nonzero value. The Movie Toolbox calls your error-notification function only in response to errors generated by the Movie Toolbox.

Special Considerations

Error-notification functions can be especially useful when you are debugging your program. The Movie Toolbox manages the sticky error value.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Movies.h`

SetMovieTime

Changes a movie's current time.

```
void SetMovieTime (
   Movie theMovie,
   const TimeRecord *newtime
);
```

Parameters

`theMovie`

The movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie` (page 108), `NewMovieFromFile`, and `NewMovieFromHandle`.

`newtime`

A pointer to a `TimeRecord` structure containing the new time.

Return Value
You can access error returns from this function through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71). See `Error Codes`.

Discussion
The Movie Toolbox saves the movie's current time when you save the movie.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
QTPixelBufferVCToCGImage

Declared In
Movies.h

SetMovieTimeScale

Establishes a movie's time scale.

```
ComponentResult ADD_MEDIA_BASENAME() SetMovieTimeScale
```

Parameters

theMovie

The movie for this operation. Your application obtains this movie identifier from such functions as NewMovie (page 108), NewMovieFromFile, and NewMovieFromHandle.

timeScale

The movie's new time scale.

Return Value
You can access error returns from this function through GetMoviesError (page 70) and GetMoviesStickyError (page 71). See Error Codes.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
MakeEffectMovie
makeeffectslideshow
makeeffectslideshow.win
vrmakepano
vrmakepano.win

Declared In
Movies.h

SetMovieTimeValue

Sets a movie's time value.

```
void SetMovieTimeValue (
   Movie theMovie,
   TimeValue newtime
);
```

Parameters

theMovie

The movie for this operation. Your application obtains this movie identifier from such functions as NewMovie (page 108), NewMovieFromFile, and NewMovieFromHandle.

newtime

The new time value. You must ensure that the time value is in the movie's time scale.

Return Value
You can access error returns from this function through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71). See `Error Codes`.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
CompressMovies
DigitizerShell
DragAndDrop Shell
MovieGWorlds
QT Internals

Declared In
`Movies.h`

SetMovieVideoOutput

Indicates to the ICM the video output component being used with a given movie.

```
void SetMovieVideoOutput (
   Movie theMovie,
   ComponentInstance vout
);
```

Parameters

`theMovie`

The movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie` (page 108), `NewMovieFromFile`, and `NewMovieFromHandle`.

`vout`

The video output component. Applications obtain this reference from `OpenComponent` or `OpenDefaultComponent`. Call the function and pass `NIL` in this parameter as soon as the video output component is no longer in use.

Discussion
As soon as you turn on the echo port on any video output component, you should make this call so the ICM keeps track of the video output in use.

Version Notes
Introduced in QuickTime 5.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Movies.h`

SetMovieVisualContext

Targets a movie to render into a visual context.

```
OSStatus SetMovieVisualContext (
   Movie movie,
   QTVisualContextRef visualContext
);
```

Parameters

movie

The movie.

visualContext

The visual context that the movie will render into. May be NULL..

Return Value

An error code. Returns `noErr` if there is no error. Returns `memFullErr` if memory cannot be allocated. Returns `kQTVisualContextNotAllowed` if the movie is not able to render using a visual context. Returns `paramErr` if the movie is NULL.

Discussion

When `SetMovieVisualContext` (page 151) succeeds, it will retain the `QTVisualContext` object for its own use. If `visualContext` is NULL, the movie will not render any visual media.
`SetMovieVisualContext` (page 151) will fail if a different movie is already using the visual context, so you should first `disassociate` the other movie by calling `SetMovieVisualContext` (page 151) with a NULL `visualContext`.

Availability

Available in Mac OS X v10.3 and later.

Related Sample Code

QTCoreImage101
QTCoreVideo102
QTCoreVideo103
QTCoreVideo201
QTCoreVideo301

Declared In

`Movies.h`

SetMovieVolume

Sets a movie's current volume but does not store the setting in the movie.

```
void SetMovieVolume (
   Movie theMovie,
   short volume
);
```

Parameters

theMovie

The movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie` (page 108), `NewMovieFromFile`, and `NewMovieFromHandle`.

volume

The current volume setting of the movie represented as a 16-bit, fixed-point number. The high-order 8 bits contain the integer part of the value; the low-order 8 bits contain the fractional part. Volume values range from -1.0 to 1.0. Negative values play no sound but preserve the absolute value of the volume setting. You can use the constants shown below. See these constants:

Return Value
You can access error returns from this function through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71). See `Error Codes`.

Discussion
The setting made by this function is not persistent. To store a volume setting in the movie, call `SetMoviePreferredVolume` (page 143).

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
MovieBrowser
QTMusicToo
SimpleCocoaMovie
vrscript
vrscript.win

Declared In
`Movies.h`

SetTimeBaseFlags

Sets the contents of the control flags of a time base.

```
void SetTimeBaseFlags (
   TimeBase tb,
   long timeBaseFlags
);
```

Parameters

tb

The time base for this operation. Your application obtains this time base identifier from `NewTimeBase` (page 110).

timeBaseFlags

The control flags for this time base (see below). You may set only one flag to 1. Be sure to set unused flags to 0. See these constants:

`loopTimeBase`

`palindromeLoopTimeBase`

Return Value
You can access error returns from this function through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71). See `Error Codes`.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
MakeEffectMovie
vrmovies
vrmovies.win
vrscript
vrscript.win

Declared In
`Movies.h`

SetTimeBaseMasterClock

Assigns a clock component to a time base.

```
void SetTimeBaseMasterClock (
   TimeBase slave,
   Component clockMeister,
   const TimeRecord *slaveZero
);
```

Parameters

slave
The time base for this operation. Your application obtains this time base identifier from `NewTimeBase` (page 110).

clockMeister
The clock component to be assigned to this time base. Your application can obtain this component identifier from `FindNextComponent`.

slaveZero
A pointer to the time, in the clock's time scale, that corresponds to a 0 time value for the slave time base. This parameter allows you to set an offset between the time base and the clock component. Set this parameter to `NIL` if there is no offset.

Return Value
You can access error returns from this function through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71). See `Error Codes`.

Discussion
A time base derives its time from either a clock component or from another time base. Don't use this function to assign a clock to a movie's time base.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
BrideOfMungGrab
QTMusicToo

Declared In
`Movies.h`

SetTimeBaseMasterTimeBase

Assigns a master time base to a time base.

```
void SetTimeBaseMasterTimeBase (
   TimeBase slave,
   TimeBase master,
   const TimeRecord *slaveZero
);
```

Parameters

slave

The time base for this operation. Your application obtains this time base identifier from `NewTimeBase` (page 110).

master

The master time base to be assigned to this time base. Your application obtains this time base identifier from `NewTimeBase` (page 110).

slaveZero

A pointer to the time, in the time scale of the master time base, that corresponds to a 0 time value for the slave time scale. This parameter allows you to set an offset between the time base and the master time base. Set this parameter to `NIL` if there is no offset.

Return Value
You can access error returns from this function through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71). See `Error Codes`.

Discussion
A time base derives its time from either a clock component or another time base. Don't use this function to assign a master time base to a movie's time base.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
LiveVideoMixer2
LiveVideoMixer3
QTMusicToo

Declared In
`Movies.h`

SetTimeBaseOffsetTimeBase

Attaches an offset time base to another time base.

```
OSErr SetTimeBaseOffsetTimeBase (
   TimeBase tb,
   TimeBase offsettb,
   const TimeRecord *offsetZero
);
```

Parameters

masterOffsetTimeBase

The time base to which the offset time base is to be attached. A `NIL` value can be passed when the offset time base has already be set but a new offset value is needed.

offsetTimeBase

The offset time base to be attached.

offsetZero

A pointer to a `TimeRecord` value set to the offset between the master time base and the offset time base. Passing a negative value means the offset time base will start sooner.

Return Value

See `Error Codes` in the QuickTime API Reference. Returns `noErr` if there is no error.

Version Notes

Introduced in QuickTime 6.4.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`Movies.h`

SetTimeBaseRate

Sets the rate of a time base.

```
void SetTimeBaseRate (
   TimeBase tb,
   Fixed r
);
```

Parameters

tb

The time base for this operation. Your application obtains this time base identifier from `NewTimeBase` (page 110).

r

The rate of the time base.

Return Value

You can access error returns from this function through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71). See `Error Codes`.

Discussion

Rates may be set to negative values. Negative rates cause time to move backward for the time base.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
qteffects.win
QTMusicToo
qtshoweffect
VideoProcessing
vrscript.win

Declared In
`Movies.h`

SetTimeBaseStartTime

Sets the start time of a time base.

```
void SetTimeBaseStartTime (
   TimeBase tb,
   const TimeRecord *tr
);
```

Parameters

tb

The time base for this operation. Your application obtains this time base identifier from `NewTimeBase` (page 110).

tr

A pointer to a `TimeRecord` structure that contains the start time value.

Return Value
You can access error returns from this function through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71). See `Error Codes`.

Discussion
The start time defines the time base's minimum time value. You must specify the new start time in a time structure.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Movies.h`

SetTimeBaseStopTime

Sets the stop time of a time base.

```
void SetTimeBaseStopTime (
   TimeBase tb,
   const TimeRecord *tr
);
```

Parameters

tb

The time base for this operation. Your application obtains this time base identifier from `NewTimeBase` (page 110).

tr

A pointer to a `TimeRecord` structure that contains the stop time value.

Return Value
You can access error returns from this function through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71). See `Error Codes`.

Discussion
The stop time defines the time base's maximum time value. You must specify the new stop time in a time structure.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Movies.h`

SetTimeBaseTime

Sets the current time of a time base.

```
void SetTimeBaseTime (
   TimeBase tb,
   const TimeRecord *tr
);
```

Parameters

tb

The time base for this operation. Your application obtains this time base identifier from `NewTimeBase` (page 110).

tr

A pointer to a `TimeRecord` structure that contains the current time value.

Return Value
You can access error returns from this function through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71). See `Error Codes`.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
Movies.h

SetTimeBaseValue

Sets the current time of a time base.

```
void SetTimeBaseValue (
   TimeBase tb,
   TimeValue t,
   TimeScale s
);
```

Parameters

tb

The time base for this operation. Your application obtains this time base identifier from NewTimeBase (page 110).

t

The new time value.

s

The time scale of the new time value.

Return Value
You can access error returns from this function through GetMoviesError (page 70) and GetMoviesStickyError (page 71). See Error Codes.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
qteffects.win
qtshoweffect
Show Movie
VideoProcessing
vrscript.win

Declared In
Movies.h

SetTimeBaseZero

Changes the offset from a time base to either its master time base or its clock component.

```
void SetTimeBaseZero (
   TimeBase tb,
   TimeRecord *zero
);
```

Parameters

tb

The time base for this operation. Your application obtains this time base identifier from `NewTimeBase` (page 110).

zero

A pointer to the time that corresponds to a 0 time value for the slave time scale. This parameter allows you to set an offset between the time base and its time source. Set this parameter to `NIL` if there is no offset.

Return Value

You can access error returns from this function through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71). See `Error Codes`.

Discussion

You establish the initial offset when you assign the time base to its time source.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Movies.h`

SetTrackClipRgn

Sets the clipping region of a track.

```
void SetTrackClipRgn (
   Track theTrack,
   RgnHandle theClip
);
```

Parameters

theTrack

The track for this operation. Your application obtains this track identifier from such functions as `NewMovieTrack` and `GetMovieTrack`.

theClip

A handle to the track's clipping region. The Movie Toolbox makes a copy of this region. Your application must dispose of the region referred to by this parameter when you are done with it. Set this parameter to `NIL` to disable clipping for the track.

Return Value

You can access error returns from this function through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71). See `Error Codes`.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
AlwaysPreview

Declared In
`Movies.h`

SetTrackGWorld

Forces a track to draw into a particular graphics world, which may be different from that of the movie.

```
void SetTrackGWorld (
   Track theTrack,
   CGrafPtr port,
   GDHandle gdh,
   TrackTransferUPP proc,
   long refCon
);
```

Parameters

theTrack

The track for this operation. Your application obtains this track identifier from such functions as `NewMovieTrack` and `GetMovieTrack`.

port

Points to the graphics port structure or graphics world to which to draw the `track`. Set this parameter to `NIL` to use the movie's graphics port.

gdh

A handle to the movie's graphics device structure. Set this parameter to `NIL` to use the current device. If the `port` parameter specifies a graphics world, set this parameter to `NIL` to use that graphics world's graphics device.

proc

A pointer to your `TrackTransferProc` callback. Set this parameter to `NIL` if you want to remove your callback.

refCon

A value to pass to your `TrackTransferProc` callback. Use this parameter to point to a data structure containing any information your callback needs.

Return Value
You can access error returns from this function through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71). See `Error Codes`.

Discussion
After this function draws a track, it calls your transfer callback to copy the track to the actual movie graphics world. When your transfer callback is called, the current graphics world is set to the correct destination. You can also install a transfer callback and set the graphics world to `NIL`. In this case, the function calls your callback only as a notification that the track has been drawn; no transfer needs to take place.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
MovieGWorlds

Declared In
`Movies.h`

SetTrackMatte

Sets a track's matte.

```
void SetTrackMatte (
   Track theTrack,
   PixMapHandle theMatte
);
```

Parameters

theTrack

The track for this operation. Your application obtains this track identifier from such functions as `NewMovieTrack` and `GetMovieTrack`.

theMatte

A handle to the matte. The Movie Toolbox makes a copy of the matte, including its `ColorTable` structure and pixels. Consequently, your application must dispose of the matte when you are done with it. Set this parameter to `NIL` to remove the track's matte.

Return Value
You can access error returns from this function through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71). See `Error Codes`.

Discussion
This matte defines which of the track's pixels are displayed in a movie. You must specify the matte in a `PixMap` structure. The Movie Toolbox displays the weighted average of the track and its destination based on the corresponding pixel in the matte.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
Inside Mac Movie TB Code

Declared In
`Movies.h`

ShowMoviePoster

Displays a movie's poster.

```
void ShowMoviePoster (
   Movie theMovie
);
```

Parameters

theMovie

A movie identifier. Your application obtains this identifier from such functions as `NewMovie` (page 108), `NewMovieFromFile`, and `NewMovieFromHandle`.

Return Value

You can access error returns from this function through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71). See `Error Codes`.

Discussion

The Movie Toolbox draws the movie poster once, in the movie's graphics world, using the movie's matrix and display clipping characteristics. You can access error returns from this function through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71). See `Error Codes`.

Special Considerations

This function works on both active and inactive movies.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Movies.h`

SpriteMediaCountImages

Retrieves the number of images that currently exist in a sprite track.

```
ComponentResult SpriteMediaCountImages (
   MediaHandler mh,
   short *numImages
);
```

Parameters

mh

The sprite media handler for this operation.

numImages

A pointer to a short integer. On return, this integer contains the number of images for the sprite media's current time.

Return Value

You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Discussion

This function determines the number of images that currently exist based on the key frame that is in effect.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Movies.h`

SpriteMediaCountSprites

Retrieves the number of sprites that currently exist in a sprite track.

```
ComponentResult SpriteMediaCountSprites (
   MediaHandler mh,
   short *numSprites
);
```

Parameters

mh

The sprite media handler for this operation.

numSprites

A pointer to a short integer. On return, this integer contains the number of sprites for the sprite media's current time.

Return Value
You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Discussion
This function determines the number of sprites that currently exist based on the key frame that is in effect.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Movies.h`

SpriteMediaDisposeImage

Frees the memory allocated for a sprite image outside a movie and removes that image from the sprite track in which it appears.

```
ComponentResult SpriteMediaDisposeImage (
   MediaHandler mh,
   short imageIndex
);
```

Parameters

mh

The sprite media handler for this operation.

imageIndex

The index of a sprite image that was previously created by SpriteMediaNewImage (page 175). If you know only the image ID, you can convert it to the index by calling SpriteMediaImageIDToIndex (page 174).

Return Value
See Error Codes. Returns noErr if there is no error.

Discussion
The image disposed of is no longer available to the sprite track, and the image index location remains empty for the duration of the current key sample.

Version Notes
Introduced in QuickTime 6.

Availability
Available in Mac OS X v10.2 and later.

Declared In
Movies.h

SpriteMediaDisposeSprite

Disposes of memory allocated for a sprite.

```
ComponentResult SpriteMediaDisposeSprite (
   MediaHandler mh,
   QTAtomID spriteID
);
```

Parameters

mh

The sprite media handler for this operation.

spriteID

The ID of the sprite for this operation.

Return Value
You can access Movie Toolbox error returns through GetMoviesError (page 70) and GetMoviesStickyError (page 71), as well as in the function result. See Error Codes.

Version Notes
Introduced in QuickTime 4.

Availability
Available in Mac OS X v10.0 and later.

Declared In
Movies.h

SpriteMediaGetActionVariable

Returns the value of the sprite track variable with the specified ID.

```
ComponentResult SpriteMediaGetActionVariable (
   MediaHandler mh,
   QTAtomID variableID,
   float *value
);
```

Parameters

mh

The sprite media handler for this operation.

variableID

A variable ID of the sprite variable.

value

A pointer to a floating-point value. If the specified variable has never been set, the value is set to 0 and the error `cannotFindAtomErr` is returned.

Return Value

You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Movies.h`

SpriteMediaGetActionVariableAsString

Undocumented

```
ComponentResult SpriteMediaGetActionVariableAsString (
   MediaHandler mh,
   QTAtomID variableID,
   Handle *theCString
);
```

Parameters

mh

The sprite media handler for this operation.

variableID

A variable ID of the sprite variable.

theCString

A pointer to a handle to a C string.

Return Value

You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Version Notes

Introduced in QuickTime 4.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Movies.h`

SpriteMediaGetDisplayedSampleNumber

Retrieves the number of the sprite media sample that is currently being displayed.

```
ComponentResult SpriteMediaGetDisplayedSampleNumber (
   MediaHandler mh,
   long *sampleNum
);
```

Parameters

mh

The sprite media handler for this operation.

sampleNum

A pointer to a long integer. On return, this integer contains the number of the sample that is currently being displayed.

Return Value
You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Movies.h`

SpriteMediaGetImageName

Returns the name of the image with the specified index from the current key frame sample.

```
ComponentResult SpriteMediaGetImageName (
   MediaHandler mh,
   short imageIndex,
   Str255 imageName
);
```

Parameters

mh

The sprite media handler for this operation.

imageIndex

The index of the image whose image name is to be retrieved. This value must be between 1 and the number of available images. You can determine how many images are available by calling `SpriteMediaCountImages` (page 162).

imageName

Returns a Pascal string with the image name of the image, or an empty string if the image is unnamed.

Return Value
You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Movies.h`

SpriteMediaGetIndImageDescription

Retrieves an image description for a specified image in a sprite track.

```
ComponentResult SpriteMediaGetIndImageDescription (
   MediaHandler mh,
   short imageIndex,
   ImageDescriptionHandle imageDescription
);
```

Parameters

mh

The sprite media handler for this operation.

imageIndex

The index of the image whose image description is to be retrieved. This value must be between 1 and the number of available images. You can determine how many images are available by calling `SpriteMediaCountImages` (page 162).

imageDescription

Specifies an image description handle. On return, this handle contains the `ImageDescription` structure that describes the specified image. This handle must be unlocked; the function resizes the handle if necessary.

Return Value
You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Movies.h`

SpriteMediaGetIndImageProperty

Returns a property value for a sprite image specified by an index.

```
ComponentResult SpriteMediaGetIndImageProperty (
   MediaHandler mh,
   short imageIndex,
   long imagePropertyType,
   void *imagePropertyValue
);
```

Parameters

mh

The sprite media handler for this operation.

imageIndex

The index of the image whose property value is to be retrieved. This value must be between 1 and the number of available images. You can determine how many images are available by calling `SpriteMediaCountImages` (page 162).

imagePropertyType

The property whose value should be retrieved (see below). See these constants:

`kSpritePropertyMatrix`
`kSpritePropertyImageDescription`
`kSpritePropertyImageDataPtr`
`kSpritePropertyVisible`
`kSpritePropertyLayer`
`kSpritePropertyGraphicsMode`

imagePropertyValue

A pointer to a variable that will hold the selected property value on return.

Return Value

You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Movies.h`

SpriteMediaGetProperty

Gets a sprite property; superseded by SpriteMediaGetSpriteProperty.

```
ComponentResult SpriteMediaGetProperty (
   MediaHandler mh,
   short spriteIndex,
   long propertyType,
   void *propertyValue
);
```

Parameters

mh

The sprite media handler for this operation.

spriteIndex

The index of the sprite for this operation.

propertyType

The property whose value should be retrieved (see below). See these constants:

```
kSpritePropertyMatrix
kSpritePropertyImageDescription
kSpritePropertyImageDataPtr
kSpritePropertyVisible
kSpritePropertyLayer
kSpritePropertyGraphicsMode
```

propertyValue

A pointer to a variable that will hold the selected property value on return.

Return Value

You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Movies.h`

SpriteMediaGetSpriteActionsForQTEvent

Gets the sprite action atom for an event.

```
ComponentResult SpriteMediaGetSpriteActionsForQTEvent (
   MediaHandler mh,
   QTEventRecordPtr event,
   QTAtomID spriteID,
   QTAtomContainer *container,
   QTAtom *atom
);
```

Parameters

mh

The sprite media handler for this operation.

event
A pointer to a `QTEventRecord` structure.

spriteID
The ID of the sprite for this operation.

container
A pointer to a QT atom container.

atom
A pointer to a QT atom in the container.

Return Value
You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Movies.h`

SpriteMediaGetSpriteName

Returns the name of the sprite with the specified ID from the currently displayed sample.

```
ComponentResult SpriteMediaGetSpriteName (
   MediaHandler mh,
   QTAtomID spriteID,
   Str255 spriteName
);
```

Parameters

mh
The sprite media handler for this operation.

spriteID
The sprite ID of the sprite name.

spriteName
Returns a Pascal string with the name of the sprite or an empty string if the sprite is unnamed.

Return Value
You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Movies.h`

SpriteMediaGetSpriteProperty

Retrieves the value of the specified sprite or sprite track property.

```
ComponentResult SpriteMediaGetSpriteProperty (
   MediaHandler mh,
   QTAtomID spriteID,
   long propertyType,
   void *propertyValue
);
```

Parameters

mh

The sprite media handler for this operation.

spriteID

The ID of the sprite for this operation. Pass `'Trck'` to return the properties of a whole sprite track.

propertyType

A constant (see below) that specifies the property whose value should be retrieved. See these constants:

- `kSpritePropertyMatrix`
- `kSpritePropertyImageDescription`
- `kSpritePropertyImageDataPtr`
- `kSpritePropertyVisible`
- `kSpritePropertyLayer`
- `kSpritePropertyGraphicsMode`
- `kSpriteTrackPropertyAllSpritesHitTestingMode`
- `kSpriteTrackPropertyPreferredDepthInterpretationMode`

propertyValue

On return, a pointer to the value of the `property`; the `data` type of that value depends on the property.

Return Value

You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Version Notes

Function introduced in QuickTime 3 or earlier. The `kSpriteTrackPropertyAllSpritesHitTestingMode` and `kSpriteTrackPropertyPreferredDepthInterpretationMode` constants were added in QuickTime 6.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

MovieSprites
qtsprites.win
qtspritesplus.win
vrscript
vrscript.win

Declared In

`Movies.h`

SpriteMediaHitTestAllSprites

Determines whether any sprites are at a specified location.

```
ComponentResult SpriteMediaHitTestAllSprites (
   MediaHandler mh,
   long flags,
   Point loc,
   QTAtomID *spriteHitID
);
```

Parameters

mh

The sprite media handler for this operation.

flags

Specifies flags (see below) that control the hit testing operation. See these constants:

`spriteHitTestBounds`

`spriteHitTestImage`

`spriteHitTestInvisibleSprites`

`spriteHitTestIsClick`

`spriteHitTestLocInDisplayCoordinates`

loc

A point in the coordinate system of the sprite track's movie to test for the existence of a sprite.

spriteHitID

A pointer to a short integer. On return, this integer contains the ID of the frontmost sprite at the location specified by `loc`. If no sprite exists at the location, the function sets the `value` of this parameter to 0.

Return Value

You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Discussion

You call this function to determine whether any sprites exist at a specified location in the coordinate system of a sprite track's movie. You can pass flags to this function to control the hit testing operation more precisely. For example, you may want the hit test operation to detect a sprite whose bounding box contains the specified location.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

MovieSprites

qtsprites

qtsprites.win

qtspritesplus.win

vrscript.win

Declared In

`Movies.h`

SpriteMediaHitTestOneSprite

Performs a hit testing operation on the sprite specified by a spriteID.

```
ComponentResult SpriteMediaHitTestOneSprite (
   MediaHandler mh,
   QTAtomID spriteID,
   long flags,
   Point loc,
   Boolean *wasHit
);
```

Parameters

mh

The sprite media handler for this operation.

spriteID

The sprite ID of the sprite.

flags

Flags (see below) that control the hit testing operation. See these constants:

`spriteHitTestBounds`

`spriteHitTestImage`

`spriteHitTestInvisibleSprites`

`spriteHitTestIsClick`

`spriteHitTestLocInDisplayCoordinates`

loc

A point to test for the existence of a sprite. The point should be defined in the local coordinates of the sprite track, unless the `spriteHitTestLocInDisplayCoordinates` flag is set.

wasHit

A pointer to a Boolean. If the sprite is hit, `wasHit` is set to TRUE; otherwise, it is set to FALSE.

Return Value

You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Discussion

This routine allows you to hit test a sprite which is fully or partially covered by other sprites.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Movies.h`

SpriteMediaHitTestSprites

Undocumented

```
ComponentResult SpriteMediaHitTestSprites (
   MediaHandler mh,
   long flags,
   Point loc,
   short *spriteHitIndex
);
```

Parameters

mh

The sprite media handler for this operation.

flags

Flags (see below) that control the hit testing operation. See these constants:

```
spriteHitTestBounds
spriteHitTestImage
spriteHitTestInvisibleSprites
spriteHitTestIsClick
spriteHitTestLocInDisplayCoordinates
```

loc

A point to test for the existence of a sprite.

spriteHitIndex

Undocumented

Return Value

You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Movies.h`

SpriteMediaImageIDToIndex

Returns the index of an outside sprite image from the ID of that image.

```
ComponentResult SpriteMediaImageIDToIndex (
   MediaHandler mh,
   QTAtomID imageID,
   short *imageIndex
);
```

Parameters

mh

The sprite media handler for this operation.

imageID

The ID of a sprite image.

imageIndex
On return, a pointer to the index of the image.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Version Notes
Introduced in QuickTime 6.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`Movies.h`

SpriteMediaImageIndexToID

Returns the ID of an outside sprite image from the index of that image.

```
ComponentResult SpriteMediaImageIndexToID (
   MediaHandler mh,
   short imageIndex,
   QTAtomID *imageID
);
```

Parameters
mh
The sprite media handler for this operation.
imageIndex
The index of a sprite image.
imageID
On return, a pointer to the ID of the image.

Return Value
See `Error Codes`. Returns `noErr` if there is no error.

Version Notes
Introduced in QuickTime 6.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`Movies.h`

SpriteMediaNewImage

Creates a new movie sprite image outside a movie.

```
ComponentResult SpriteMediaNewImage (
   MediaHandler mh,
   Handle dataRef,
   OSType dataRefType,
   QTAtomID desiredID
);
```

Parameters

mh

The sprite media handler for this operation.

dataRef

A pointer to a URL or an alias that references the image to be used as a sprite image.

dataRefType

A four character code for the type of the `dataRef` parameter. See `Component Identifiers`. For example, pass `URLDataHandlerSubType` if `dataRef` is a URL

desiredID

The desired ID identifier for the image. If the requested ID is in use, the call returns an error. If you pass 0 the function assigns the next sequential integer ID, which is usually the same as the next available index unless that ID has been previously assigned.

Return Value

See `Error Codes`. Returns `noErr` if there is no error.

Discussion

The newly created image can be used in a sprite track like any other sprite image. It can be referenced by the next available image index, equal to the number of images in the track before the call was made +1, or by the ID that was requested via the `desiredID` parameter.

Version Notes

Introduced in QuickTime 6.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`Movies.h`

SpriteMediaNewSprite

Creates a new sprite.

```
ComponentResult SpriteMediaNewSprite (
   MediaHandler mh,
   QTRuntimeSpriteDescPtr newSpriteDesc
);
```

Parameters

mh

The sprite media handler for this operation.

newSpriteDesc

A pointer to a `QTRuntimeSpriteDescStruct` structure.

Return Value
You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Version Notes
Introduced in QuickTime 4.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Movies.h`

SpriteMediaSetActionVariable

Sets the value of a sprite track variable to a specified value.

```
ComponentResult SpriteMediaSetActionVariable (
   MediaHandler mh,
   QTAtomID variableID,
   const float *value
);
```

Parameters

mh
The sprite media handler for this operation.

variableID
A variable ID of the sprite name.

value
A pointer to a floating-point number. The value is passed by reference.

Return Value
You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Discussion
This function is specific to sprite tracks using wired sprites.

Special Considerations
This function is specific to sprite tracks using wired sprites.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Movies.h`

SpriteMediaSetActionVariableToString

Undocumented

```
ComponentResult SpriteMediaSetActionVariableToString (
   MediaHandler mh,
   QTAtomID variableID,
   Ptr theCString
);
```

Parameters

mh

The sprite media handler for this operation.

variableID

A variable ID of the sprite variable.

theCString

A pointer to a C string.

Return Value

You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Version Notes

Introduced in QuickTime 4.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Movies.h`

SpriteMediaSetProperty

Sets a sprite property; superseded by SpriteMediaSetSpriteProperty.

```
ComponentResult SpriteMediaSetProperty (
   MediaHandler mh,
   short spriteIndex,
   long propertyType,
   void *propertyValue
);
```

Parameters

mh

The sprite media handler for this operation.

spriteIndex

The index of the sprite for this operation.

propertyType

The property whose value should be set (see below). See these constants:

`kSpritePropertyMatrix`
`kSpritePropertyImageDescription`
`kSpritePropertyImageDataPtr`
`kSpritePropertyVisible`
`kSpritePropertyLayer`
`kSpritePropertyGraphicsMode`

propertyValue

A pointer to a variable that contains the new value of the selected property. The type of data you pass for this parameter depends on the `property` type.

Return Value
You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Movies.h`

SpriteMediaSetSpriteProperty

Sets the specified property of a sprite or sprite track.

```
ComponentResult SpriteMediaSetSpriteProperty (
   MediaHandler mh,
   QTAtomID spriteID,
   long propertyType,
   void *propertyValue
);
```

Parameters

mh

The sprite media handler for this operation.

spriteID

The ID of the sprite for this operation. Pass `'Trck'` to set the properties of a whole sprite track.

propertyType

A constant (see below) that specifies the property whose value should be set. See these constants:

`kSpritePropertyMatrix`
`kSpritePropertyImageDescription`
`kSpritePropertyImageDataPtr`
`kSpritePropertyVisible`
`kSpritePropertyLayer`
`kSpritePropertyGraphicsMode`
`kSpriteTrackPropertyAllSpritesHitTestingMode`
`kSpriteTrackPropertyPreferredDepthInterpretationMode`

propertyValue

A pointer to the new value of the selected property. The type of data you pass for this parameter depends on the `property` type.

Return Value
You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Version Notes
Function introduced in QuickTime 3 or earlier. The `kSpriteTrackPropertyAllSpritesHitTestingMode` and `kSpriteTrackPropertyPreferredDepthInterpretationMode` constants were added in QuickTime 6.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
MovieSprites
qtsprites.win
qtspritesplus.win
vrscript
vrscript.win

Declared In
`Movies.h`

SpriteMediaSpriteIDToIndex

Converts a sprite ID to the corresponding sprite index.

```
ComponentResult SpriteMediaSpriteIDToIndex (
   MediaHandler mh,
   QTAtomID spriteID,
   short *spriteIndex
);
```

Parameters

mh
The sprite media handler for this operation.

spriteID
The ID of the sprite for this operation.

spriteIndex
On return, a pointer to the index of the sprite.

Return Value
You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Movies.h`

SpriteMediaSpriteIndexToID

Returns the ID of a sprite specified by a sprite index.

```
ComponentResult SpriteMediaSpriteIndexToID (
   MediaHandler mh,
   short spriteIndex,
   QTAtomID *spriteID
);
```

Parameters

mh

The sprite media handler for this operation.

spriteIndex

The index of the sprite for this operation.

spriteID

A pointer to the sprite ID corresponding to the sprite index. If a sprite with the specified index does not exist, the error `paramErr` is returned.

Return Value

You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Movies.h`

StartMovie

Starts the movie playing from the current movie time.

```
void StartMovie (
   Movie theMovie
);
```

Parameters

theMovie

The movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie` (page 108), `NewMovieFromFile`, and `NewMovieFromHandle`.

Return Value

You can access error returns from this function through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71). See `Error Codes`.

Discussion

You are not required to call this function to start a movie. It is included in the QuickTime API for convenience. Before playing the movie, the Movie Toolbox makes the movie active, prerolls the movie, and sets the movie to its preferred playback rate. You can use `SetMoviePreferredRate` (page 142) to change this setting.

Special Considerations

A movie's current time is saved when a movie is stored in a movie file. Therefore, your application should appropriately position a movie before playing the movie. Use `GoToBeginningOfMovie` (page 89) to set a movie to play from its start.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
CarbonQTGraphicImport
Graphic Import-Export
MovieGWorlds
OpenGLCompositorLab
vrscript.win

Declared In
`Movies.h`

StopMovie

Stops the playback of a movie.

```
void StopMovie (
   Movie theMovie
);
```

Parameters

theMovie

The movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie` (page 108), `NewMovieFromFile`, and `NewMovieFromHandle`.

Return Value
You can access this function's error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71).

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
CarbonQTGraphicImport
Graphic Import-Export
vrmovies
vrscript
vrscript.win

Declared In
`Movies.h`

SubtractTime

Subtracts one time from another.

```
void SubtractTime (
   TimeRecord *dst,
   const TimeRecord *src
);
```

Parameters

dst

A pointer to a `TimeRecord` structure. This time structure contains one of the operands for the subtraction. This function returns the result of the subtraction into this time structure as a duration.

src

A pointer to a `TimeRecord` structure. The Movie Toolbox subtracts this value from the time or duration specified by the `dst` parameter.

Return Value

You can access error returns from this function through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71). See `Error Codes`.

Discussion

If the two times are relative to different time scales or time bases, this function converts the times as appropriate to yield reasonable results. However, the time bases for both time values must rely on the same time source.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Movies.h`

TextMediaAddHiliteSample

Provides dynamic highlighting of text.

```
ComponentResult TextMediaAddHiliteSample (
   MediaHandler mh,
   short hiliteStart,
   short hiliteEnd,
   RGBColor *rgbHiliteColor,
   TimeValue duration,
   TimeValue *sampleTime
);
```

Parameters

mh

The media handler for the text media obtained by `GetMediaHandler`.

hiliteStart

Indicates the beginning of the text to be highlighted.

hiliteEnd

Indicates the ending of the text to be highlighted. If the value of the `hiliteStart` parameter equals that of the `hiliteEnd` parameter, then no text is highlighted (that is, highlighting is turned off for the duration of the specified sample).

rgbHiliteColor

A pointer to the `RGBColor` structure that defines the color for highlighting. If this parameter is not `NIL`, then the specified color is used when highlighting the text indicated by the `hiliteStart` and `hiliteEnd` parameters. Otherwise, the default system highlight color is used.

duration

Specifies how long the text sample should last. This duration is expressed in the media's time base.

sampleTime

A pointer to a time value. The actual media time at which the sample was added is returned here.

Return Value

You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Movies.h`

TextMediaAddTESample

Specifies a TextEdit handle to be added to a specified media.

```
ComponentResult TextMediaAddTESample (
   MediaHandler mh,
   TEHandle hTE,
   RGBColor *backColor,
   short textJustification,
   Rect *textBox,
   long displayFlags,
   TimeValue scrollDelay,
   short hiliteStart,
   short hiliteEnd,
   RGBColor *rgbHiliteColor,
   TimeValue duration,
   TimeValue *sampleTime
);
```

Parameters

mh

The media handler for the text media obtained by `GetMediaHandler`.

hTE

A handle to a `TERec` structure.

backColor

A pointer to an `RGBColor` structure specifying the text background color. Passing `NIL` for this parameter in defaults to white.

textJustification

Indicates the justification of the text (see below). See these constants:

textBox

A pointer to a `Rect` structure that defines the box within which the text is to be displayed. The box is relative to the track bounds.

displayFlags

Contains the text display flags (see below). See these constants:

```
dfDontDisplay
dfDontAutoScale
dfClipToTextBox
dfShrinkTextBoxToFit
dfScrollIn
dfScrollOut
dfHorizScroll
dfReverseScroll
dfContinuousScroll
dfFlowHoriz
dfContinuousKaraoke
dfDropShadow
dfAntiAlias
dfKeyedText
dfInverseHilite
dfTextColorHilite
```

scrollDelay

Indicates the delay in scrolling associated with the setting of the `dfScrollIn` and `dfScrollOut` display flags. If the value of the `scrollDelay` parameter is greater than 0 and the `dfScrollIn` flag is set, the text pauses when it has scrolled all the way in for the amount of time specified by `scrollDelay`. If the `dfScrollOut` flag is set, the pause occurs first before the text scrolls out. If both these flags are set, the pause occurs at the midpoint between scrolling in and scrolling out.

hiliteStart

The beginning of the text to be highlighted.

hiliteEnd

The end of the text to be highlighted. If the `hiliteEnd` parameter is greater than the `hiliteStart` parameter, then the text is highlighted from the selection specified by `hiliteStart` to `hiliteEnd`. To specify additional highlighting, you can use `TextMediaAddHiliteSample` (page 183).

rgbHiliteColor

Contains a pointer to an `RGBColor` structure that defines the color for highlighting. If this parameter is not `NIL`, then the specified color is used when highlighting the text indicated by the `hiliteStart` and `hiliteEnd` parameters. Otherwise, the default system highlight color is used.

duration

A time value that specifies how long the text sample should last. This duration is expressed in the media's time base.

sampleTime

Contains a pointer to a time value. The actual media time at which the sample was added is returned here.

Return Value
You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Special Considerations

Be sure to turn on the `dfDropShadow` display flag after you call this function.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Movies.h`

TextMediaAddTextSample

Adds a single block of styled text to an existing media.

```
ComponentResult TextMediaAddTextSample (
   MediaHandler mh,
   Ptr text,
   unsigned long size,
   short fontNumber,
   short fontSize,
   Style textFace,
   RGBColor *textColor,
   RGBColor *backColor,
   short textJustification,
   Rect *textBox,
   long displayFlags,
   TimeValue scrollDelay,
   short hiliteStart,
   short hiliteEnd,
   RGBColor *rgbHiliteColor,
   TimeValue duration,
   TimeValue *sampleTime
);
```

Parameters

mh
The media handler for the text media obtained by `GetMediaHandler`.

text
A pointer to a block of text.

size
Indicates the size of the text block, in bytes.

fontNumber
The number for the font in which to display the text.

fontSize
Indicates the size of the font.

textFace

Indicates the typeface or `style` of the text (that is, bold, italic, and so on).

textColor

A pointer to an `RGBColor` structure specifying the color of the text. Passing `NIL` for this parameter in defaults to black.

backColor

A pointer to an `RGBColor` structure specifying the text background color. Passing `NIL` for this parameter in defaults to white.

textJustification

Indicates the justification of the text (see below). See these constants:

textBox

A pointer to a `Rect` structure that defines the box within which the text is to be displayed. The box is relative to the track bounds.

displayFlags

Contains the text display flags (see below). See these constants:

```
dfDontDisplay
dfDontAutoScale
dfClipToTextBox
dfShrinkTextBoxToFit
dfScrollIn
dfScrollOut
dfHorizScroll
dfReverseScroll
dfContinuousScroll
dfFlowHoriz
dfContinuousKaraoke
dfDropShadow
dfAntiAlias
dfKeyedText
dfInverseHilite
dfTextColorHilite
```

scrollDelay

Indicates the delay in scrolling associated with the setting of the `dfScrollIn` and `dfScrollOut` display flags. If the value of the `scrollDelay` parameter is greater than 0 and the `dfScrollIn` flag is set, the text pauses when it has scrolled all the way in for the amount of time specified by `scrollDelay`. If the `dfScrollOut` flag is set, the pause occurs first before the text scrolls out. If both these flags are set, the pause occurs at the midpoint between scrolling in and scrolling out.

hiliteStart

The beginning of the text to be highlighted.

hiliteEnd

The end of the text to be highlighted. If the `hiliteEnd` parameter is greater than the `hiliteStart` parameter, then the text is highlighted from the selection specified by `hiliteStart` to `hiliteEnd`. To specify additional highlighting, you can use `TextMediaAddHiliteSample` (page 183).

rgbHiliteColor

Contains a pointer to an `RGBColor` structure that defines the color for highlighting. If this parameter is not `NIL`, then the specified color is used when highlighting the text indicated by the `hiliteStart` and `hiliteEnd` parameters. Otherwise, the default system highlight color is used.

duration

A time value that specifies how long the text sample should last. This duration is expressed in the media's time base.

sampleTime

Contains a pointer to a time value. The actual media time at which the sample was added is returned here.

Return Value

You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Special Considerations

Be sure to turn on the `dfDropShadow` display flag after you call this function.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

addhtactions.win

qttext

qttext.win

qtwiredactions

qtwiredactions.win

Declared In

`Movies.h`

TextMediaDrawRaw

Undocumented

```
ComponentResult TextMediaDrawRaw (
   MediaHandler mh,
   GWorldPtr gw,
   GDHandle gd,
   void *data,
   long dataSize,
   TextDescriptionHandle tdh
);
```

Parameters

mh

The text media handler obtained by `GetMediaHandler`.

gw

A pointer to a `CGrafPort` structure that defines a graphics world.

gd

A handle to a graphics device.

data

A pointer to the source data.

dataSize

The size of the source data.

tdh

A handle to a `TextDescription` structure.

Return Value
You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Version Notes
Introduced in QuickTime 4.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Movies.h`

TextMediaFindNextText

Searches for text with a specified media handler starting at a given time.

```
ComponentResult TextMediaFindNextText (
   MediaHandler mh,
   Ptr text,
   long size,
   short findFlags,
   TimeValue startTime,
   TimeValue *foundTime,
   TimeValue *foundDuration,
   long *offset
);
```

Parameters

mh

The media handler for the text media obtained by `GetMediaHandler`.

text

Points to the text to be found.

size

The length of the text to be found.

findFlags

Flags (see below) that determine the conditions of the search. See these constants:

`findTextEdgeOK`
`findTextCaseSensitive`
`findTextReverseSearch`
`findTextWrapAround`
`findTextUseOffset`

startTime

Indicates the time (expressed in the movie time scale) at which to begin the search.

foundTime

A pointer to the movie time at which the text sample is found if the search is successful. Otherwise, it returns -1.

foundDuration

A pointer to the duration of the sample (in the movie time scale) that is found if the search is successful.

offset

A pointer to the offset of the found text from the beginning of the text portion of the sample.

Return Value

You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

qttext

qttext.win

Declared In

`Movies.h`

TextMediaGetTextProperty

Sets properties of a text media.

```
ComponentResult TextMediaGetTextProperty (
   MediaHandler mh,
   TimeValue atMediaTime,
   long propertyType,
   void *data,
   long dataSize
);
```

Parameters

mh

The text media handler obtained by `GetMediaHandler`.

atMediaTime

The media time of the text.

propertyType

A constant (see below) that identifies the text property to be set. See these constants:

```
kTextTextHandle
kTextTextPtr
kTextTEStyle
kTextBackColor
kTextForeColor
kTextFace
kTextFont
kTextSize
kTextAlignment
kTextHilite
kTextDropShadow
kTextDisplayFlags
kTextScroll
```

data

A pointer to text data.

dataSize

The size of the data.

Return Value

You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Version Notes

Introduced in QuickTime 5.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Movies.h`

TextMediaHiliteTextSample

Specifies selected text to be highlighted for a given text media handler.

```
ComponentResult TextMediaHiliteTextSample (
   MediaHandler mh,
   TimeValue sampleTime,
   short hiliteStart,
   short hiliteEnd,
   RGBColor *rgbHiliteColor
);
```

Parameters

mh

The text media handler obtained by `GetMediaHandler`.

sampleTime

The starting time in the sample.

hiliteStart
The beginning of the text to be highlighted.

hiliteEnd
The end of the text to be highlighted. If the `hiliteEnd` parameter is greater than the `hiliteStart` parameter, then the text is highlighted from the selection specified by `hiliteStart` to `hiliteEnd`.

rgbHiliteColor
Contains a pointer to an `RGBColor` structure that defines the color for highlighting. If this parameter is not `NIL`, then the specified color is used when highlighting the text indicated by the `hiliteStart` and `hiliteEnd` parameters. Otherwise, the default system highlight color is used.

Return Value
You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
qttext
qttext.win

Declared In
`Movies.h`

TextMediaRawIdle

Undocumented

```
ComponentResult TextMediaRawIdle (
   MediaHandler mh,
   GWorldPtr gw,
   GDHandle gd,
   TimeValue sampleTime,
   long flagsIn,
   long *flagsOut
);
```

Parameters

mh
The text media handler obtained by `GetMediaHandler`.

gw
A pointer to a `CGrafPort` structure that defines a graphics world.

gd
A handle to a graphics device.

sampleTime
Undocumented

flagsIn
Undocumented

flagsOut

Undocumented

Return Value
You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Version Notes
Introduced in QuickTime 4.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Movies.h`

TextMediaRawSetup

Undocumented

```
ComponentResult TextMediaRawSetup (
   MediaHandler mh,
   GWorldPtr gw,
   GDHandle gd,
   void *data,
   long dataSize,
   TextDescriptionHandle tdh,
   TimeValue sampleDuration
);
```

Parameters

mh

The text media handler obtained by `GetMediaHandler`.

gw

A pointer to a `CGrafPort` structure that defines a graphics world.

gd

A handle to a graphics device.

data

A pointer to data.

dataSize

The size of the data.

tdh

A handle to a `TextDescription` structure.

sampleDuration

Undocumented

Return Value
You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Version Notes
Introduced in QuickTime 4.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Movies.h`

TextMediaSetTextProc

Specifies a custom function to be called whenever a text sample is displayed in a movie.

```
ComponentResult TextMediaSetTextProc (
   MediaHandler mh,
   TextMediaUPP TextProc,
   long refcon
);
```

Parameters

mh

The text media handler obtained by `GetMediaHandler`.

TextProc

A Universal Procedure Pointer that points to a `TextMediaProc` callback.

refcon

Indicates a reference constant that will be passed to your callback. Use this parameter to point to a data structure containing any information your function needs. Set this parameter to 0 if you don't need it.

Return Value
You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
qttext
qttext.win

Declared In
`Movies.h`

TextMediaSetTextProperty

Sets properties of a text media.

```
ComponentResult TextMediaSetTextProperty (
   MediaHandler mh,
   TimeValue atMediaTime,
   long propertyType,
   void *data,
   long dataSize
);
```

Parameters

mh

The text media handler obtained by `GetMediaHandler`.

atMediaTime

The media time of the text.

propertyType

A constant (see below) that identifies the text property to be set. See these constants:

`kTextTextHandle`
`kTextTextPtr`
`kTextTEStyle`
`kTextBackColor`
`kTextForeColor`
`kTextFace`
`kTextFont`
`kTextSize`
`kTextAlignment`
`kTextHilite`
`kTextDropShadow`
`kTextDisplayFlags`
`kTextScroll`

data

A pointer to text data.

dataSize

The size of the data.

Return Value

You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Version Notes

Introduced in QuickTime 4.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Movies.h`

TextMediaSetTextSampleData

Sets values before calling TextMediaAddTextSample or TextMediaAddTESample.

```
ComponentResult TextMediaSetTextSampleData (
   MediaHandler mh,
   void *data,
   OSType dataType
);
```

Parameters

mh

A reference to the text media handler. You obtain this reference from `GetMediaHandler`.

data

A pointer to the data, defined by the `dataType` parameter.

dataType

The type of data.

Return Value

You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Discussion

The following code sample demonstrates how to use this function:

```
// TextMediaSetTextSampleData coding example
short           trans =127;
Point           dropOffset;
MediaHandler    mh;
dropOffset.h =dropOffset.v =4
TextMediaSetTextSampleData(mh,(void *)&dropOffset,dropShadowOffsetType);
TextMediaSetTextSampleData(mh,(void *)&trans,dropShadowTranslucencyType);
```

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Movies.h`

UpdateMovie

Ensures that the Movie Toolbox properly displays your movie after it has been uncovered.

```
OSErr UpdateMovie (
   Movie theMovie
);
```

Parameters

theMovie

The movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie` (page 108), `NewMovieFromFile`, and `NewMovieFromHandle`.

Return Value

You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Discussion
Your application should call this function during window updating. Don't call `MoviesTask` (page 106) at this time; you will observe better display behavior if you call it at the end of your update processing.

This function does not actually update the movie's graphics world. Rather, it invalidates the movie's display state so that the Movie Toolbox redraws the movie the next time you call `MoviesTask`. If you need to force a movie to be redrawn outside of a window update sequence, your application can call this function and then call `MoviesTask` to service the movie. The Movie Toolbox determines the portion of the screen to update by examining the graphics port's visible region.

The following code snippet uses this function in a Macintosh Window Manager update sequence:

```
// UpdateMovie coding example
#include <Events.h>
#include <ToolUtils.h>
#include "Movies.h"
void DoUpdate (WindowRef theWindow, Movie theMovie)
{
    BeginUpdate (theWindow);
    UpdateMovie (theMovie);
    EndUpdate (theWindow);
} /* DoUpdate */
```

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
CompressMovies
DigitizerShell
DragAndDrop Shell
mfc.win
MovieGWorlds

Declared In
`Movies.h`

VideoMediaGetCodecParameter

Undocumented

```
ComponentResult VideoMediaGetCodecParameter (
   MediaHandler mh,
   CodecType cType,
   OSType parameterID,
   Handle outParameterData
);
```

Parameters

mh

A reference to a video media handler. You obtain this reference from `GetMediaHandler`.

cType
A valid codec type constant; see `Codec Identifiers`.

parameterID
Undocumented

outParameterData
A handle to the returned data.

Return Value
You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Version Notes
Introduced in QuickTime 4.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Movies.h`

VideoMediaGetStallCount

Undocumented

```
ComponentResult VideoMediaGetStallCount (
   MediaHandler mh,
   unsigned long *stalls
);
```

Parameters

mh
A reference to a video media handler. You obtain this reference from `GetMediaHandler`.

stalls
The number of stalls.

Return Value
You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Version Notes
Introduced in QuickTime 3 or earlier.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Movies.h`

VideoMediaGetStatistics

Returns the play-back frame rate of a movie.

```
ComponentResult VideoMediaGetStatistics (
   MediaHandler mh
);
```

Parameters

mh

A reference to a video media handler. You obtain this reference from `GetMediaHandler`.

Return Value

The average frame rate since the last time `VideoMediaResetStatistics` (page 199) was called. Because of sampling errors, the values returned from this function are accurate only after waiting at least one second after calling `VideoMediaResetStatistics`.

Discussion

This function can only be used on video or MPEG media handlers. Because not all QuickTime movies have a constant frame rate, the results of this call can be difficult to interpret correctly. For this reason, the results of this function should not be displayed in a place where a novice user is likely to see it.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Movies.h`

VideoMediaResetStatistics

Resets the video media handler's counters before using VideoMediaGetStatistics to determine the frame rate of a movie.

```
ComponentResult VideoMediaResetStatistics (
   MediaHandler mh
);
```

Parameters

mh

A reference to a video media handler. You obtain this reference from `GetMediaHandler`.

Return Value

You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Special Considerations

This call can only be used on video or MPEG media handlers.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Movies.h`

VideoMediaSetCodecParameter

Undocumented

```
ComponentResult VideoMediaSetCodecParameter (
   MediaHandler mh,
   CodecType cType,
   OSType parameterID,
   long parameterChangeSeed,
   void *dataPtr,
   long dataSize
);
```

Parameters

mh

A reference to a video media handler. You obtain this reference from `GetMediaHandler`.

cType

A valid codec type constant; see `Codec Identifiers`.

parameterID

Undocumented

parameterChangeSeed

Undocumented

dataPtr

A pointer to the data to be set.

dataSize

The size of the data.

Return Value

You can access Movie Toolbox error returns through `GetMoviesError` (page 70) and `GetMoviesStickyError` (page 71), as well as in the function result. See `Error Codes`.

Version Notes

Introduced in QuickTime 4.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Movies.h`

Callbacks

Data Types

ControlPtr

Represents a type used by the Movie Manager API.

```
typedef ControlRecord * ControlPtr;
```

Availability
Available in Mac OS X v10.0 through Mac OS X v10.4.

Declared In
`Controls.h`

ControlRef

Represents a type used by the Movie Manager API.

```
typedef ControlPtr * ControlRef;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`HIObject.h`

QTFloatSingle

Represents a type used by the Movie Manager API.

```
typedef Float32 QTFloatSingle;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Movies.h`

QTNewMoviePropertyElement

Stores a movie property for NewMovieFromProperties.

```
struct QTNewMoviePropertyElement {
QTPropertyClass       propClass;
QTPropertyID          propID;
ByteCount             propValueSize;
QTPropertyValuePtr    propValueAddress;
OSStatus              propStatus;
};
```

Fields
`propClass`

Discussion
A four-character code designating the class of a movie property. See `New Movie Property Codes`.

`propID`

Discussion
The ID of the property.

`propValueSize`

Discussion
The size in bytes of the property passed in `propValueAddress`.

`propValueAddress`

Discussion
A pointer to a movie property. Since the `data` type is fixed for each element's property class and ID, these is no ambiguity about the `data` type for its property value.

`propStatus`

Discussion
Indicates any problems with the property. For example, if a property is not understood by the function it is passed to, this field is set appropriately. See the discussion in `NewMovieFromProperties` (page 109).

Discussion
When you call `NewMovieFromProperties` (page 109), you allocate and own arrays of these elements to pass to it, as well as the property values that each element points to. You are responsible for disposing of all of these memory allocations.

Related Functions
Associated function: `NewMovieFromProperties` (page 109)

Declared In
`Movies.h`

QTRuntimeSpriteDescPtr

Represents a type used by the Movie Manager API.

```
typedef QTRuntimeSpriteDescStruct * QTRuntimeSpriteDescPtr;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Movies.h`

QTRuntimeSpriteDescStruct

Provides a sprite description for the SpriteMediaNewSprite function.

```
QTRuntimeSpriteDescStruct {
     long                                version;
     QTAtomID                            spriteID;
     short                               imageIndex;
     MatrixRecord                        matrix;
     short                               visible;
     short                               layer;
     ModifierTrackGraphicsModeRecord     graphicsMode;
     QTAtomID                            actionHandlingSpriteID;
 };
```

Fields

`version`

Discussion
Set to 0.

`spriteID`

Discussion
The QT atom ID of the sprite atom.

`imageIndex`

Discussion
The index of the sprite image. This value must be between 1 and the number of available images. You can determine how many images are available by calling `SpriteMediaCountImages` (page 162).

`matrix`

Discussion
A `MatrixRecord` structure that defines the sprite's matrix.

`visible`

Discussion
Undocumented

`layer`

Discussion
The sprite's layer number.

`graphicsMode`

Discussion
A `ModifierTrackGraphicsModeRecord` structure that defines the graphics mode setting for the sprite.

`actionHandlingSpriteID`

Discussion
Undocumented

Declared In
`Movies.h`

RegionCode

Represents a type used by the Movie Manager API.

```
typedef SInt16 RegionCode;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`MacTypes.h`

Style

Represents a type used by the Movie Manager API.

```
typedef unsigned char Style;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`MacTypes.h`

TEHandle

Represents a type used by the Movie Manager API.

```
typedef TEPtr * TEHandle;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`TextEdit.h`

TEPtr

Represents a type used by the Movie Manager API.

```
typedef TERec * TEPtr;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`TextEdit.h`

TextDescriptionHandle

Represents a type used by the Movie Manager API.

```
typedef TextDescriptionPtr * TextDescriptionHandle;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
Movies.h

TextDescriptionPtr

Represents a type used by the Movie Manager API.

```
typedef TextDescription * TextDescriptionPtr;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
Movies.h

TimeBaseStatus

Represents a type used by the Movie Manager API.

```
typedef unsigned long TimeBaseStatus;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
Movies.h

Constants

TextMediaFindNextText Values

Constants passed to TextMediaFindNextText.

```
enum {
  findTextEdgeOK                = 1 << 0, /* Okay to find text at specified sample
 time*/
  findTextCaseSensitive         = 1 << 1, /* Case sensitive search*/
  findTextReverseSearch         = 1 << 2, /* Search from sampleTime backwards*/
  findTextWrapAround            = 1 << 3, /* Wrap search when beginning or end of
 movie is hit*/
  findTextUseOffset             = 1 << 4 /* Begin search at the given character
offset into sample rather than edge*/
};
```

Declared In
Movies.h

QTTextToNativeText Values

Constants passed to QTTextToNativeText.

```
enum {
  kITextAtomType                = 'itxt',
  kITextStringAtomType          = 'text'
};
```

Declared In
Movies.h

ITextRemoveString Values

Constants passed to ITextRemoveString.

```
enum {
  kITextRemoveEverythingBut     = 0 << 1,
  kITextRemoveLeaveSuggestedAlternate = 1 << 1
};
```

Declared In
Movies.h

CreateMovieControl Values

Constants passed to CreateMovieControl.

```
enum {
  kMovieControlOptionHideController = (1L << 0),
  kMovieControlOptionLocateTopLeft = (1L << 1),
  kMovieControlOptionEnableEditing = (1L << 2),
  kMovieControlOptionHandleEditingHI = (1L << 3),
  kMovieControlOptionSetKeysEnabled = (1L << 4),
  kMovieControlOptionManuallyIdled = (1L << 5)
};
```

Declared In
Movies.h

MovieMediaGetCurrentMovieProperty Values

Constants passed to MovieMediaGetCurrentMovieProperty.

```
enum {
  kMoviePropertyDuration        = 'dura', /* TimeValue **/
  kMoviePropertyTimeScale       = 'tims', /* TimeValue **/
  kMoviePropertyTime            = 'timv', /* TimeValue **/
  kMoviePropertyNaturalBounds   = 'natb', /* Rect **/
  kMoviePropertyMatrix          = 'mtrx', /* Matrix **/
  kMoviePropertyTrackList       = 'tlst' /* long ****/
};
```

Declared In
Movies.h

EnterMoviesOnThread Values

Constants passed to EnterMoviesOnThread.

```
enum {
  kQTEnterMoviesFlagDontSetComponentsThreadMode = 1L << 0
};
```

Declared In
Movies.h

loopTimeBase

Constants grouped with loopTimeBase.

```
enum {
  loopTimeBase                  = 1,
  palindromeLoopTimeBase        = 2,
  maintainTimeBaseZero          = 4
};
```

Declared In
Movies.h

SetMovieDrawingCompleteProc Values

Constants passed to SetMovieDrawingCompleteProc.

```
enum {
  movieDrawingCallWhenChanged   = 0,
  movieDrawingCallAlways        = 1
};
```

Declared In
Movies.h

timeBaseAfterStopTime

Constants grouped with timeBaseAfterStopTime.

```
enum {
  timeBaseBeforeStartTime       = 1,
  timeBaseAfterStopTime         = 2,
  timeBaseRateChanging          = 4
};
```

Declared In
`Movies.h`

Document Revision History

This table describes the changes to *Movie Manager Reference*.

Date	Notes
2006-12-14	Correct minor typos.
2006-05-23	New document, based on previously published material, that describes the API for the QuickTime Movie Manager.

Document Revision History

Index

I

L

M

N

P

Q

R

S

T

U

V