Media Types and Media Handlers Reference

QuickTime > Media Types & Media Handlers

2006-11-10

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

Simultaneously published in the United States and Canada.

Contents

Media Types and Media Handlers Reference 7

Overview 7
Functions by Task 7
- General Data Management 7
- Managing Graphics Data 8
- Managing Media Chunks 9
- Managing Your Media Handler Component 9
- Sound Media Handler Functions 9
- Supporting Keyboard Focus 10
- Video Media Handler Functions 10
- Working With The Idle Manager 10
- Supporting Functions 10

Functions 13
- CallComponentExecuteWiredAction 13
- DisposePrePrerollCompleteUPP 13
- MediaChangedNonPrimarySource 14
- MediaCompare 14
- MediaCurrentMediaQueuedData 15
- MediaDisposeTargetRefCon 16
- MediaDoIdleActions 16
- MediaEmptyAllPurgeableChunks 17
- MediaEmptySampleCache 17
- MediaEnterEmptyEdit 18
- MediaFlushNonPrimarySourceData 18
- MediaForceUpdate 19
- MediaGetActionsForQTEvent 19
- MediaGetChunkManagementFlags 20
- MediaGetClock 21
- MediaGetDrawingRgn 21
- MediaGetEffectiveSoundBalance 22
- MediaGetEffectiveVolume 22
- MediaGetErrorString 23
- MediaGetGraphicsMode 24
- MediaGetInvalidRegion 24
- MediaGetMediaInfo 25
- MediaGetMediaLoadState 25
- MediaGetName 26
- MediaGetNextBoundsChange 27
- MediaGetNextStepTime 28
- MediaGetOffscreenBufferSize 28
- MediaGetPublicInfo 29

MediaGetPurgeableChunkMemoryAllowance 30
MediaGetSampleDataPointer 31
MediaGetSoundBalance 31
MediaGetSoundBassAndTreble 32
MediaGetSoundEqualizerBandLevels 33
MediaGetSoundEqualizerBands 33
MediaGetSoundLevelMeterInfo 34
MediaGetSoundLevelMeteringEnabled 34
MediaGetSoundOutputComponent 35
MediaGetSrcRgn 35
MediaGetTrackOpaque 36
MediaGetURLLink 37
MediaGetUserPreferredCodecs 38
MediaGetVideoParam 38
MediaGGetIdleManager 39
MediaGGetStatus 40
MediaGSetActiveSegment 40
MediaGSetIdleManager 41
MediaGSetVolume 42
MediaHasCharacteristic 42
MediaHitTestForTargetRefCon 43
MediaHitTestTargetRefCon 44
MediaIdle 45
MediaInitialize 47
MediaInvalidateRegion 48
MediaMakeMediaTimeTable 48
MediaMCIsPlayerEvent 50
MediaNavigateTargetRefCon 50
MediaPrePrerollBegin 51
MediaPrePrerollCancel 52
MediaPreroll 52
MediaPutMediaInfo 53
MediaQueueNonPrimarySourceData 54
MediaRefConGetProperty 55
MediaRefConSetProperty 56
MediaReleaseSampleDataPointer 56
MediaResolveTargetRefCon 57
MediaSampleDescriptionB2N 58
MediaSampleDescriptionChanged 58
MediaSampleDescriptionN2B 59
MediaSetActionsCallback 59
MediaSetActive 60
MediaSetChunkManagementFlags 61
MediaSetClip 61
MediaSetDimensions 62
MediaSetDoMCActionCallback 63

MediaSetGraphicsMode 63
MediaSetGWorld 64
MediaSetHandlerCapabilities 65
MediaSetHints 66
MediaSetMatrix 67
MediaSetMediaTimeScale 67
MediaSetMovieTimeScale 68
MediaSetNonPrimarySourceData 69
MediaSetPublicInfo 71
MediaSetPurgeableChunkMemoryAllowance 72
MediaSetRate 72
MediaSetScreenLock 73
MediaSetSoundBalance 74
MediaSetSoundBassAndTreble 74
MediaSetSoundEqualizerBands 75
MediaSetSoundLevelMeteringEnabled 76
MediaSetSoundLocalizationData 76
MediaSetSoundOutputComponent 77
MediaSetTrackInputMapReference 77
MediaSetUserPreferredCodecs 78
MediaSetVideoParam 79
MediaTargetRefConsEqual 80
MediaTimeBaseChanged 81
MediaTrackEdited 81
MediaTrackPropertyAtomChanged 82
MediaTrackReferencesChanged 82
MediaVideoOutputChanged 83
NewPrePrerollCompleteUPP 83
Callbacks 84
PrePrerollCompleteProc 84
Data Types 84
GetMovieCompleteParams 84
LevelMeterInfo 87
LevelMeterInfoPtr 88
MediaEQSpectrumBandsRecord 88
MediaEQSpectrumBandsRecordPtr 88
PrePrerollCompleteUPP 89
QTCustomActionTargetPtr 89
QTCustomActionTargetRecord 89
Constants 90
MediaForceUpdate Values 90
Data Handler Flags 91
MediaSetChunkManagementFlags Values 91
MediaSetVideoParam Values 91
MediaNavigateTargetRefCon Values 92
MediaRefConSetProperty Values 92

Media Task Flags 92
MediaHitTestTargetRefCon Values 93

Document Revision History 95

Index 97

Media Types and Media Handlers Reference

Framework: Frameworks/QuickTime.framework

Declared in MediaHandlers.h
Sound.h

Overview

QuickTime media handler components interpret and manipulate media types, such as sound, video, music, text, timecodes, and tweens.

Functions by Task

General Data Management

`MediaCompare` (page 14)
Lets a media handler determine whether the Movie Toolbox should allow one track to be pasted into another.

`MediaGetMediaInfo` (page 25)
Lets a derived media handler obtain the private data stored in its media.

`MediaGetName` (page 26)
Returns the name of the media type.

`MediaGetNextStepTime` (page 28)
Searches for the next forward or backward step time from the given media time.

`MediaGetOffscreenBufferSize` (page 28)
Determines the dimensions of the offscreen buffer.

`MediaGetSampleDataPointer` (page 31)
Allows a derived media handler to obtain a pointer to the sample data for a particular sample number, the size of that sample, and the index of the sample description associated with that sample.

`MediaGetVideoParam` (page 38)
Retrieves the value of the brightness, contrast, hue, sharpness, saturation, black level, or white level of a video image.

`MediaGSetActiveSegment` (page 40)
Informs your derived media handlers of the current active segment.

`MediaHasCharacteristic` (page 42)
Called by Movie Toolbox with a specified characteristic to allow tracks to be identified by various attributes.

`MediaInvalidateRegion` (page 48)
Updates the invalidated display region the next time MediaIdle is called.

`MediaPreroll` (page 52)
Prepares a media handler for playback.

`MediaPutMediaInfo` (page 53)
Lets a derived media handler store proprietary information in its media.

`MediaReleaseSampleDataPointer` (page 56)
Balances calls to MediaGetSampleDataPointer to release allocated memory.

`MediaSampleDescriptionChanged` (page 58)
Informs a media handler that SetMediaSampleDescription has been called for a specified sample description.

`MediaSetActive` (page 60)
Enables and disables media.

`MediaSetHints` (page 66)
Implements the appropriate behavior for the various media hints such as scrub mode and high-quality mode.

`MediaSetMediaTimeScale` (page 67)
Informs a media handler that its media's time scale has been changed.

`MediaSetMovieTimeScale` (page 68)
Informs a media handler that the movie's time scale has been changed.

`MediaSetNonPrimarySourceData` (page 69)
Allows a media handler to support receiving media data from other media handlers.

`MediaSetRate` (page 72)
Sets a media's playback rate.

`MediaSetTrackInputMapReference` (page 77)
Provides a derived media handler with an updated input map.

`MediaSetVideoParam` (page 79)
Lets you dynamically adjust the brightness, contrast, hue, sharpness, saturation, black level, and white level of a video image.

`MediaTrackEdited` (page 81)
Informs a derived media handler about edits to its track.

`MediaTrackPropertyAtomChanged` (page 82)
Notifies the derived media handler whenever its media property atom has changed.

`MediaTrackReferencesChanged` (page 82)
Notifies the derived media handler whenever the track references in the movie change.

Managing Graphics Data

`MediaGetDrawingRgn` (page 21)
Specifies a portion of the screen that must be redrawn, defined in the movie's display coordinate system.

`MediaGetNextBoundsChange` (page 27)
Determines when a media causes a spatial change to a movie.

`MediaGetSrcRgn` (page 35)
Specifies an irregular destination display region to the Movie Toolbox.

`MediaGetTrackOpaque` (page 36)
Determines whether a media is transparent or opaque when displayed.

`MediaSetClip` (page 61)
Specifies changes to a derived media handler's clipping region.

`MediaSetDimensions` (page 62)
Informs a media handler when its media's spatial dimensions change.

`MediaSetGWorld` (page 64)
Lets a derived media handler learn about changes to its media's graphic environment.

`MediaSetMatrix` (page 67)
Tells a media handler about changes to either the movie matrix or the track matrix.

Managing Media Chunks

`MediaEmptyAllPurgeableChunks` (page 17)
Force QuickTime to empty all purgeable media chunks in this application.

`MediaGetChunkManagementFlags` (page 20)
Returns the current settings of the media chunk management flags.

`MediaGetPurgeableChunkMemoryAllowance` (page 30)
Returns the current purgeable chunk memory allowance.

`MediaSetChunkManagementFlags` (page 61)
Sets application-global flags that control media chunk management.

`MediaSetPurgeableChunkMemoryAllowance` (page 72)
Sets the maximum amount of memory that QuickTime will allow purgeable chunks to occupy.

Managing Your Media Handler Component

`MediaGGetStatus` (page 40)
Reports error conditions to the Movie Toolbox.

`MediaIdle` (page 45)
Provides processing time to a derived media handler during movie playback.

`MediaInitialize` (page 47)
Prepares a derived media handler component to provide access to its media.

Sound Media Handler Functions

`MediaGetSoundBalance` (page 31)
Obtains the right/left sound balance of a track.

`MediaSetSoundBalance` (page 74)
Sets the right/left sound balance of a track.

Supporting Keyboard Focus

MediaNavigateTargetRefCon (page 50)
Locates the object for keyboard focus.

MediaRefConGetProperty (page 55)
Returns the current media handler state based on the property type.

MediaRefConSetProperty (page 56)
Sets a new media handler state based on the property type.

Video Media Handler Functions

MediaGetGraphicsMode (page 24)
Obtains the graphics mode and blend color values currently in use by any media handler.

MediaSetGraphicsMode (page 63)
Sets the graphics mode and blend color of any media handler.

Working With The Idle Manager

MediaGGetIdleManager (page 39)
Retrieves an Idle Manager object from a derived media handler.

MediaGSetIdleManager (page 41)
Lets a derived media handler report its idling needs.

Supporting Functions

CallComponentExecuteWiredAction (page 13)
Undocumented

DisposePrePrerollCompleteUPP (page 13)
Disposes of a PrePrerollCompleteUPP pointer.

MediaChangedNonPrimarySource (page 14)
Informs a media handler of a change in the source of media data from another media handler.

MediaCurrentMediaQueuedData (page 15)
Retrieves the timing of the current media in queued data.

MediaDisposeTargetRefCon (page 16)
Disposes any resources allocated as part of calling MediaHitTestForTargetRefCon.

MediaDoIdleActions (page 16)
Forces a media handler to perform its idle-time actions.

MediaEmptySampleCache (page 17)
Deletes any sample data that the media handler has cached.

MediaEnterEmptyEdit (page 18)
Undocumented

MediaFlushNonPrimarySourceData (page 18)
Flushes data that a media handler gets from another media handler.

`MediaForceUpdate` (page 19)
Forces a media update.

`MediaGetActionsForQTEvent` (page 19)
Returns an event handler for your media handler.

`MediaGetClock` (page 21)
Gets the clock component associated with a media.

`MediaGetEffectiveSoundBalance` (page 22)
Gets the effective sound balance setting of a media handler.

`MediaGetEffectiveVolume` (page 22)
Gets the effective volume setting for a media handler.

`MediaGetErrorString` (page 23)
Undocumented

`MediaGetInvalidRegion` (page 24)
Gets the invalid region for a media handler's current display.

`MediaGetMediaLoadState` (page 25)
Queried by GetMovieLoadState to help determine a movie's load state.

`MediaGetPublicInfo` (page 29)
Undocumented

`MediaGetSoundBassAndTreble` (page 32)
Gets the bass and treble settings for a media handler.

`MediaGetSoundEqualizerBandLevels` (page 33)
Gets the sound equalizer band levels for a media handler.

`MediaGetSoundEqualizerBands` (page 33)
Gets the sound equalizer settings for a media handler.

`MediaGetSoundLevelMeterInfo` (page 34)
Gets the right and left sound level meter values for a media handler.

`MediaGetSoundLevelMeteringEnabled` (page 34)
Determines if a media handler's sound level metering capability is enabled.

`MediaGetSoundOutputComponent` (page 35)
Gets the sound output component associated with a media handler.

`MediaGetURLLink` (page 37)
Undocumented

`MediaGetUserPreferredCodecs` (page 38)
Retrieves the list of components last passed to the media handler by a call to MediaSetUserPreferredCodecs.

`MediaGSetVolume` (page 42)
Specifies changes to the sound volume setting.

`MediaHitTestForTargetRefCon` (page 43)
Locates an object for hit testing.

`MediaHitTestTargetRefCon` (page 44)
Detects if the mouse click and its release are in the same location and within the object.

`MediaMakeMediaTimeTable` (page 48)
Called by the base media handler to create a media time table.

MediaMCIsPlayerEvent (page 50)
Undocumented

MediaPrePrerollBegin (page 51)
Undocumented

MediaPrePrerollCancel (page 52)
Cancels a media handler pre-preroll operation that was started by MediaPrePrerollBegin.

MediaQueueNonPrimarySourceData (page 54)
Undocumented

MediaResolveTargetRefCon (page 57)
Undocumented

MediaSampleDescriptionB2N (page 58)
Undocumented

MediaSampleDescriptionN2B (page 59)
Undocumented

MediaSetActionsCallback (page 59)
Sets an ActionsProc callback for a media handler.

MediaSetDoMCActionCallback (page 63)
Sets a DoMCActionProc callback for a media handler.

MediaSetHandlerCapabilities (page 65)
Lets a derived media handler report its capabilities to the base media handler.

MediaSetPublicInfo (page 71)
Undocumented

MediaSetScreenLock (page 73)
Locks the display screen for a media handler.

MediaSetSoundBassAndTreble (page 74)
Sets the bass and treble controls for a media handler.

MediaSetSoundEqualizerBands (page 75)
Sets sound equalizer bands for a media handler.

MediaSetSoundLevelMeteringEnabled (page 76)
Enables or disables sound level metering for a media handler.

MediaSetSoundLocalizationData (page 76)
Supports 3D sound capabilities in a media handler that plays sound.

MediaSetSoundOutputComponent (page 77)
Sets the sound output component for a media handler.

MediaSetUserPreferredCodecs (page 78)
Requests that a media handler favor specified codec components when selecting components with which to play media.

MediaTargetRefConsEqual (page 80)
Undocumented

MediaTimeBaseChanged (page 81)
Undocumented

MediaVideoOutputChanged (page 83)
Undocumented

`NewPrePrerollCompleteUPP` (page 83)
Allocates a Universal Procedure Pointer for the PrePrerollCompleteProc callback.

Functions

CallComponentExecuteWiredAction

Undocumented

```
ComponentResult CallComponentExecuteWiredAction (
   ComponentInstance ci,
   QTAtomContainer actionContainer,
   QTAtom actionAtom,
   QTCustomActionTargetPtr target,
   QTEventRecordPtr event
);
```

Parameters

ci
A component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

actionContainer
A QT atom container that contains the action atom.

actionAtom
The action atom for this wired action.

target
A pointer to a `QTCustomActionTargetRecord` structure.

event
A pointer to a `QTEventRecord` structure.

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

Version Notes
Introduced in QuickTime 4.

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

Declared In
`MediaHandlers.h`

DisposePrePrerollCompleteUPP

Disposes of a PrePrerollCompleteUPP pointer.

```
void DisposePrePrerollCompleteUPP (
   PrePrerollCompleteUPP userUPP
);
```

Parameters

userUPP

A `PrePrerollCompleteUPP` pointer. See `Universal Procedure Pointers`.

Return Value
You can access this function's error returns through `GetMoviesError` and `GetMoviesStickyError`.

Version Notes
Introduced in QuickTime 4.1.

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

Declared In
`MediaHandlers.h`

MediaChangedNonPrimarySource

Informs a media handler of a change in the source of media data from another media handler.

```
ComponentResult MediaChangedNonPrimarySource (
   MediaHandler mh,
   long inputIndex
);
```

Parameters

mh

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

inputIndex

The ID of the entry in the media's input map to which the changed data corresponds.

Return Value
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.

Declared In
`MediaHandlers.h`

MediaCompare

Lets a media handler determine whether the Movie Toolbox should allow one track to be pasted into another.

```
ComponentResult MediaCompare (
   MediaHandler mh,
   Boolean *isOK,
   Media srcMedia,
   ComponentInstance srcMediaComponent
);
```

Parameters

mh

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

isOK

A pointer to a Boolean value. Your media handler must set this value to TRUE if the source media and the media associated with the media handler have equivalent media settings, so that pasting the two together would cause no media information loss.

srcMedia

The source media for this operation.

srcMediaComponent

The source media component for this operation.

Return Value
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.

Declared In
`MediaHandlers.h`

MediaCurrentMediaQueuedData

Retrieves the timing of the current media in queued data.

```
ComponentResult MediaCurrentMediaQueuedData (
   MediaHandler mh,
   long *milliSecs
);
```

Parameters

mh

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

milliSecs

A pointer to the number of milliseconds to the current data.

Return Value
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.

Declared In
`MediaHandlers.h`

MediaDisposeTargetRefCon

Disposes any resources allocated as part of calling MediaHitTestForTargetRefCon.

```
ComponentResult MediaDisposeTargetRefCon (
   MediaHandler mh,
   long targetRefCon
);
```

Parameters

mh

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

targetRefCon

A reference constant set by the media handler in a call to `MediaHitTestForTargetRefCon` (page 43).

Return Value
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.

Declared In
`MediaHandlers.h`

MediaDoIdleActions

Forces a media handler to perform its idle-time actions.

```
ComponentResult MediaDoIdleActions (
   MediaHandler mh
);
```

Parameters

mh

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

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

Version Notes
Introduced in QuickTime 4.

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

Related Sample Code
SurfaceVertexProgram

Declared In
`MediaHandlers.h`

MediaEmptyAllPurgeableChunks

Force QuickTime to empty all purgeable media chunks in this application.

```
ComponentResult MediaEmptyAllPurgeableChunks (
   MediaHandler mh
);
```

Parameters

mh

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

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

Version Notes
Introduced in QuickTime 6. Can be used only with Mac OS X 10.1 and later.

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

Declared In
`MediaHandlers.h`

MediaEmptySampleCache

Deletes any sample data that the media handler has cached.

```
ComponentResult MediaEmptySampleCache (
   MediaHandler mh,
   long sampleNum,
   long sampleCount
);
```

Parameters

mh

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

sampleNum

The ID of the first sample to delete.

sampleCount

The number of samples to delete. Passing -1 means delete `sampleNum` and all samples after it.

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

Discussion
This is an optional media handler function. Most developers will not need to call it.

Version Notes
Introduced in QuickTime 5.

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

Declared In
`MediaHandlers.h`

MediaEnterEmptyEdit

Undocumented

```
ComponentResult MediaEnterEmptyEdit (
   MediaHandler mh
);
```

Parameters

mh

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

Return Value
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.

Declared In
`MediaHandlers.h`

MediaFlushNonPrimarySourceData

Flushes data that a media handler gets from another media handler.

```
ComponentResult MediaFlushNonPrimarySourceData (
   MediaHandler mh,
   long inputIndex
);
```

Parameters

mh

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

inputIndex

The ID of the entry in the media's input map to which the flushed data corresponds.

Return Value
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.

Declared In
`MediaHandlers.h`

MediaForceUpdate

Forces a media update.

```
ComponentResult MediaForceUpdate (
   MediaHandler mh,
   long forceUpdateFlags
);
```

Parameters

mh

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

forceUpdateFlags

Flags (see below) that define the update to be forced. See these constants:

`forceUpdateRedraw`

`forceUpdateNewBuffer`

Return Value
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.

Declared In
`MediaHandlers.h`

MediaGetActionsForQTEvent

Returns an event handler for your media handler.

```
ComponentResult MediaGetActionsForQTEvent (
   MediaHandler mh,
   QTEventRecordPtr event,
   long targetRefCon,
   QTAtomContainer *container,
   QTAtom *atom
);
```

Parameters

mh

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

event

A pointer to a `QTEventRecord` structure.

targetRefCon

A reference constant set by the media handler in `MediaHitTestForTargetRefCon` (page 43).

container

An atom container that you can pass back to the standard controller used for implementing sprite actions.

atom

An atom you can pass back to the standard controller used for implementing sprite actions.

Return Value

See `Error Codes`. Returns `qtEventWasHandledErr` if the event was handled by the media handler. 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.

Declared In

`MediaHandlers.h`

MediaGetChunkManagementFlags

Returns the current settings of the media chunk management flags.

```
ComponentResult MediaGetChunkManagementFlags (
   MediaHandler mh,
   UInt32 *flags
);
```

Parameters

mh

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

flags

A pointer to the constants (see below) that were set by a previous call to `MediaSetChunkManagementFlags` (page 61). See these constants:

`kEmptyPurgableChunksOverAllowance`

Return Value

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

Discussion

Do not call this function under QuickTime 5. It could cause a crash.

Version Notes

Introduced in QuickTime 6. Can be used only with Mac OS X 10.1 and later.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`MediaHandlers.h`

MediaGetClock

Gets the clock component associated with a media.

```
ComponentResult MediaGetClock (
   MediaHandler mh,
   ComponentInstance *clock
);
```

Parameters

mh

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

clock

A pointer to a clock component.

Return Value
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.

Declared In
`MediaHandlers.h`

MediaGetDrawingRgn

Specifies a portion of the screen that must be redrawn, defined in the movie's display coordinate system.

```
ComponentResult MediaGetDrawingRgn (
   MediaHandler mh,
   RgnHandle *partialRgn
);
```

Parameters

mh

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

partialRgn

A pointer to a handle to a `MacRegion` structure that defines the screen region to be redrawn, using the movie's display coordinate system. Note that your component is responsible for disposing of this region once drawing is complete. Since the base media handler will use this region during redrawing, it is best to dispose of it when your component is closed.

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

Discussion
The Movie Toolbox calls this function in order to determine what part of the screen needs to be redrawn. By default, `theMovie` Toolbox redraws the entire region that belongs to your component. If your component determines that only a portion of the screen has changed, and has indicated this to the toolbox by setting

the `mPartialDraw` flag to 1 in the `flagsOut` parameter of the `MediaIdle` (page 45) function, the toolbox calls your component's `MediaGetDrawingRgn` (page 21) function. Your component returns a region that defines the changed portion of the track's display region.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`MediaHandlers.h`

MediaGetEffectiveSoundBalance

Gets the effective sound balance setting of a media handler.

```
ComponentResult MediaGetEffectiveSoundBalance (
   MediaHandler mh,
   short *balance
);
```

Parameters

mh

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

balance

A pointer to an integer. The Movie Toolbox returns the current balance setting of the media handler as a 16-bit, fixed-point value. The high-order 8 bits contain the integer part of the value; the low-order 8 bits contain the fractional part. Valid balance values range from -1.0 to 1.0. Negative values emphasize the left sound channel, and positive values emphasize the right sound channel; a value of 0 specifies neutral balance.

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

Version Notes
Introduced in QuickTime 4.

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

Declared In
`MediaHandlers.h`

MediaGetEffectiveVolume

Gets the effective volume setting for a media handler.

```
ComponentResult MediaGetEffectiveVolume (
   MediaHandler mh,
   short *volume
);
```

Parameters

mh

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

volume

The media's current volume setting. This value is represented as a 16-bit, fixed-point number. The high-order 8 bits contain the integer portion; 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.

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

Version Notes
Introduced in QuickTime 4.

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

Declared In
`MediaHandlers.h`

MediaGetErrorString

Undocumented

```
ComponentResult MediaGetErrorString (
   MediaHandler mh,
   ComponentResult theError,
   Str255 errorString
);
```

Parameters

mh

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

theError

An error identifier; see `Error Codes`.

errorString

A text string that describes the error.

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

Version Notes
Introduced in QuickTime 4.

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

Declared In
MediaHandlers.h

MediaGetGraphicsMode

Obtains the graphics mode and blend color values currently in use by any media handler.

```
ComponentResult MediaGetGraphicsMode (
   MediaHandler mh,
   long *mode,
   RGBColor *opColor
);
```

Parameters

mh

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

mode

A pointer to a long integer. The media handler returns the graphics mode currently in use by the media handler; see Graphics Transfer Modes.

opColor

A pointer to an RGBColor structure. The Movie Toolbox returns the color currently in use by the media handler. This is the blend value for blends and the transparent color for transparent operations. The toolbox supplies this value to QuickDraw when you draw in addPin, subPin, blend, transparent, or graphicsModeStraightAlphaBlend mode.

Return Value
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.

Declared In
MediaHandlers.h

MediaGetInvalidRegion

Gets the invalid region for a media handler's current display.

```
ComponentResult MediaGetInvalidRegion (
   MediaHandler mh,
   RgnHandle rgn
);
```

Parameters

mh

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

rgn

A handle to a MacRegion structure that defines an invalid region.

Return Value
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.

Declared In
`MediaHandlers.h`

MediaGetMediaInfo

Lets a derived media handler obtain the private data stored in its media.

```
ComponentResult MediaGetMediaInfo (
   MediaHandler mh,
   Handle h
);
```

Parameters

mh

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

h

A handle to storage containing your media handler's proprietary information. Your media handler creates this private data when the Movie Toolbox calls your `MediaPutMediaInfo` (page 53) function. Do not dispose of this handle; it is owned by the Movie Toolbox.

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

Discussion
Your derived media handler should support this function if you store private data in your media.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`MediaHandlers.h`

MediaGetMediaLoadState

Queried by GetMovieLoadState to help determine a movie's load state.

```
ComponentResult MediaGetMediaLoadState (
   MediaHandler mh,
   long *mediaLoadState
);
```

Parameters

mh

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

mediaLoadState

Undocumented

Return Value

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

Discussion

`GetMovieLoadState` queries any idling importers associated with a movie, checks if the movie is fast starting, and queries media handlers. The minimum load state of all of these is then considered to be the load state of the movie.

Version Notes

Introduced in QuickTime 4.1.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`MediaHandlers.h`

MediaGetName

Returns the name of the media type.

```
ComponentResult MediaGetName (
   MediaHandler mh,
   Str255 name,
   long requestedLanguage,
   long *actualLanguage
);
```

Parameters

mh

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

name

The name of the `media` type; for example, the video media handler returns the string `'video'`.

requestedLanguage

The language in which you want the name returned; see `Localization Codes`.

actualLanguage

A pointer to the actual language in which the name is returned; see `Localization Codes`

Return Value

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
MakeEffectMovie
Movie From DataRef
qteffects.win
vrbackbuffer
vrmovies.win

Declared In
`MediaHandlers.h`

MediaGetNextBoundsChange

Determines when a media causes a spatial change to a movie.

```
ComponentResult MediaGetNextBoundsChange (
   MediaHandler mh,
   TimeValue *when
);
```

Parameters

mh

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

when

A pointer to a movie time value, which your media handler must set. Be sure to use the movie's time base. Use the current effective rate to determine the direction your media is playing. Set this value to -1 if there are no more changes in the specified direction.

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

Discussion
Your derived media handler should support this function if you change the shape of your media's spatial representation during playback. The Movie Toolbox calls this function only if you have set the `handlerHasSpatial` flag to 1 in the `flags` parameter of `MediaSetHandlerCapabilities` (page 65).

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`MediaHandlers.h`

MediaGetNextStepTime

Searches for the next forward or backward step time from the given media time.

```
ComponentResult MediaGetNextStepTime (
   MediaHandler mh,
   short flags,
   TimeValue mediaTimeIn,
   TimeValue *mediaTimeOut,
   Fixed rate
);
```

Parameters

mh

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

flags

Flags (see below) that specify search parameters. See these constants:

`nextTimeStep`

mediaTimeIn

A time value that establishes the starting point for the search. This time value is in the media's time scale.

mediaTimeOut

The step time (the time of the next frame) calculated by the media handler. The media handler should return the first time value it finds that meets the search criteria specified in the `flags` parameter. This time value is in the media's time scale.

rate

The search direction. Negative values search backward from the starting point specified in the `mediaTimeIn` parameter. Other values cause a forward search.

Return Value

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

Discussion

This function allows a derived media handler to return the next step time from the specified media time. The mechanism in QuickTime used for stepping backwards and forwards a frame at a time are the interesting time calls: `GetMovieNextInterestingTime`, `GetTrackNextInterestingTime`, and `GetMediaNextInterestingTime`.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`MediaHandlers.h`

MediaGetOffscreenBufferSize

Determines the dimensions of the offscreen buffer.

```
ComponentResult MediaGetOffscreenBufferSize (
   MediaHandler mh,
   Rect *bounds,
   short depth,
   CTabHandle ctab
);
```

Parameters

mh

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

bounds

A `Rect` structure that defines the boundaries of your offscreen buffer.

depth

The depth of the offscreen.

ctab

A handle to the `ColorTable` structure associated with the offscreen buffer.

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

Discussion
Before the base media handler allocates an offscreen buffer for your derived media handler, it calls this function. The depth and color table used for the buffer are also passed. When this function is called, the `bounds` parameter specifies the size that the base media handler intends to use for your offscreen buffer. You can modify this as appropriate before returning. This capability is useful if your media handler can draw only at particular sizes. It is also useful for implementing antialiased drawing; you can request a buffer that is larger than your destination area and have the base media handler scale the image down for you.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`MediaHandlers.h`

MediaGetPublicInfo

Undocumented

```
ComponentResult MediaGetPublicInfo (
   MediaHandler mh,
   OSType infoSelector,
   void *infoDataPtr,
   Size *ioDataSize
);
```

Parameters

mh

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

infoSelector
Undocumented

infoDataPtr
Undocumented

ioDataSize
Undocumented

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

Version Notes
Introduced in QuickTime 5.

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

Related Sample Code
qtskins
qtskins.win

Declared In
`MediaHandlers.h`

MediaGetPurgeableChunkMemoryAllowance

Returns the current purgeable chunk memory allowance.

```
ComponentResult MediaGetPurgeableChunkMemoryAllowance (
   MediaHandler mh,
   Size *allowance
);
```

Parameters

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

allowance
A pointer to the allowance in bytes.

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

Version Notes
Introduced in QuickTime 6. Can be used only with Mac OS X 10.1 and later.

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

Declared In
`MediaHandlers.h`

MediaGetSampleDataPointer

Allows a derived media handler to obtain a pointer to the sample data for a particular sample number, the size of that sample, and the index of the sample description associated with that sample.

```
ComponentResult MediaGetSampleDataPointer (
   MediaHandler mh,
   long sampleNum,
   Ptr *dataPtr,
   long *dataSize,
   long *sampleDescIndex
);
```

Parameters

mh

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

sampleNum

The number of the sample that is to be loaded.

dataPtr

A pointer to a pointer to receive the address of the loaded sample data.

dataSize

A pointer to a field that is to receive the size, in bytes, of the sample.

sampleDescIndex

A pointer to a long integer. This function returns an index value to the sample description that corresponds to the returned sample data. If you do not want this information, set the parameter to `NIL`.

Return Value

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

Discussion

This function returns a pointer to the data for a particular sample number from a movie data file. It provides access to the base media handler's caching services for sample data. It is a service provided by the base media handler for its clients.

Special Considerations

Each call to this function must be balanced by a call to `MediaReleaseSampleDataPointer` (page 56) or the memory will not be released. This function generally provides better overall performance than `GetMediaSample`.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`MediaHandlers.h`

MediaGetSoundBalance

Obtains the right/left sound balance of a track.

```
ComponentResult MediaGetSoundBalance (
   MediaHandler mh,
   short *balance
);
```

Parameters

mh

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

balance

On return, a pointer to the balance setting for the media handler's track. The balance setting is a signed 16-bit integer that controls the relative volume of the left and right sound channels. A value of 0 sets the balance to neutral. Positive values shift the balance to the right channel, negative values to the left channel. The valid range is 127 (right channel only) to -128 (left channel only).

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

Discussion
Use this function to get the audio balance on a per-track basis. It works with both mono and stereo sources. The actual volume of the audio is not changed by this setting, only its relative distribution. This function operates on sound tracks, music tracks, and Flash tracks containing audio. It may operate on other audio track types as well; if the the chosen media handler does not support this function, it returns `badComponentSelector`. To obtain the media handler for a track, call `GetTrackMedia` and then `GetMediaHandler`.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`MediaHandlers.h`

MediaGetSoundBassAndTreble

Gets the bass and treble settings for a media handler.

```
ComponentResult MediaGetSoundBassAndTreble (
   MediaHandler mh,
   short *bass,
   short *treble
);
```

Parameters

mh

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

bass

Undocumented

treble

Undocumented

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

Version Notes
Introduced in QuickTime 4.

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

Declared In
`MediaHandlers.h`

MediaGetSoundEqualizerBandLevels

Gets the sound equalizer band levels for a media handler.

```
ComponentResult MediaGetSoundEqualizerBandLevels (
   MediaHandler mh,
   UInt8 *bandLevels
);
```

Parameters

mh

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

bandLevels

Undocumented

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

Version Notes
Introduced in QuickTime 4.

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

Related Sample Code
sndequalizer
SurfaceVertexProgram

Declared In
`MediaHandlers.h`

MediaGetSoundEqualizerBands

Gets the sound equalizer settings for a media handler.

```
ComponentResult MediaGetSoundEqualizerBands (
   MediaHandler mh,
   MediaEQSpectrumBandsRecordPtr spectrumInfo
);
```

Parameters

mh

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

spectrumInfo
A pointer to a `MediaEQSpectrumBandsRecord` structure.

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

Version Notes
Introduced in QuickTime 4.

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

Declared In
`MediaHandlers.h`

MediaGetSoundLevelMeterInfo

Gets the right and left sound level meter values for a media handler.

```
ComponentResult MediaGetSoundLevelMeterInfo (
   MediaHandler mh,
   LevelMeterInfoPtr levelInfo
);
```

Parameters

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

levelInfo
A pointer to a `LevelMeterInfo` structure.

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

Version Notes
Introduced in QuickTime 4.

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

Declared In
`MediaHandlers.h`

MediaGetSoundLevelMeteringEnabled

Determines if a media handler's sound level metering capability is enabled.

```
ComponentResult MediaGetSoundLevelMeteringEnabled (
   MediaHandler mh,
   Boolean *enabled
);
```

Parameters

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

enabled

A pointer to a Boolean; it is TRUE if sound level metering is enabled, FALSE otherwise.

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

Version Notes
Introduced in QuickTime 4.

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

Declared In
`MediaHandlers.h`

MediaGetSoundOutputComponent

Gets the sound output component associated with a media handler.

```
ComponentResult MediaGetSoundOutputComponent (
   MediaHandler mh,
   Component *outputComponent
);
```

Parameters

mh

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

outputComponent

An instance of a sound output component.

Return Value
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.

Declared In
`MediaHandlers.h`

MediaGetSrcRgn

Specifies an irregular destination display region to the Movie Toolbox.

```
ComponentResult MediaGetSrcRgn (
   MediaHandler mh,
   RgnHandle rgn,
   TimeValue atMediaTime
);
```

Parameters

mh

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

rgn

A handle to a `MacRegion` structure. When the Movie Toolbox calls your function, this region is initialized to the track's boundary rectangle, which is defined by the `width` and height fields in the `GetMovieCompleteParams` structure that you obtain when the Movie Toolbox calls your `MediaInitialize` (page 47) function. Your media handler may then alter this region as appropriate, so that it corresponds to the boundaries of your media's display image. Note that this region is in the track's coordinate system, not the movie's. Do not dispose of this region; it is owned by the Movie Toolbox.

atMediaTime

The time value at which the Movie Toolbox wants to know what the source region is.

Return Value

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

Discussion

Your derived media handler should support this function if your media does not completely fill the track rectangle during playback. The Movie Toolbox calls this function only if you have set the `handlerHasSpatial` flag to 1 in the `flags` parameter of the `MediaSetHandlerCapabilities` (page 65) function.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`MediaHandlers.h`

MediaGetTrackOpaque

Determines whether a media is transparent or opaque when displayed.

```
ComponentResult MediaGetTrackOpaque (
   MediaHandler mh,
   Boolean *trackIsOpaque
);
```

Parameters

mh

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

trackIsOpaque

A pointer to a Boolean value. Your media handler must set this value to TRUE if your media is semitransparent (that is, you draw in blend mode); otherwise, leave the flag unchanged.

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

Discussion
Your derived media handler should support this function if your media is semitransparent when displayed or if you handle display transfer modes. The Movie Toolbox calls this function only if you have set the `handlerHasSpatial` or `handlerCanTransferMode` flag to 1 in the `flags` parameter of the `MediaSetHandlerCapabilities` (page 65) function.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`MediaHandlers.h`

MediaGetURLLink

Undocumented

```
ComponentResult MediaGetURLLink (
   MediaHandler mh,
   Point displayWhere,
   Handle *urlLink
);
```

Parameters

mh

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

displayWhere

Undocumented

urlLink

Undocumented

Return Value
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.

Declared In
`MediaHandlers.h`

MediaGetUserPreferredCodecs

Retrieves the list of components last passed to the media handler by a call to MediaSetUserPreferredCodecs.

```
ComponentResult MediaGetUserPreferredCodecs (
   MediaHandler mh,
   CodecComponentHandle *userPreferredCodecs
);
```

Parameters

mh

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

userPreferredCodecs

A pointer to a handle containing component identifiers. If the media handler currently has a preferred component list, it will copy that list into a new handle and store the new handle in this variable. If the media handler does not currently have a preferred component list, it will store `NIL` in this variable. The caller must dispose of this handle.

Return Value

See `Error Codes`. Returns `badComponentSelector` if the media handler component does not support this call. Returns `noErr` if there is no error.

Version Notes

Introduced in QuickTime 5.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`MediaHandlers.h`

MediaGetVideoParam

Retrieves the value of the brightness, contrast, hue, sharpness, saturation, black level, or white level of a video image.

```
ComponentResult MediaGetVideoParam (
   MediaHandler mh,
   long whichParam,
   unsigned short *value
);
```

Parameters

mh

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

whichParam

A constant (see below) that specifies the video parameter whose value you want to retrieve. See these constants:

- kMediaVideoParamBrightness
- kMediaVideoParamContrast
- kMediaVideoParamHue
- kMediaVideoParamSharpness
- kMediaVideoParamSaturation
- kMediaVideoParamBlackLevel
- kMediaVideoParamWhiteLevel

value

The actual value of the `requested` video parameter. The meaning of the values vary depending on the implementation.

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

Discussion
This function and `MediaSetVideoParam` (page 79) are currently used by the MPEG media handler.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`MediaHandlers.h`

MediaGGetIdleManager

Retrieves an Idle Manager object from a derived media handler.

```
ComponentResult MediaGGetIdleManager (
   MediaHandler mh,
   IdleManager *pim
);
```

Parameters

mh

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

pim

A pointer to a pointer to an opaque data structure that belongs to the Mac OS Idle Manager. You can get the pointer this parameter points to by calling `QTIdleManagerOpen`.

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

Discussion
This routine must be implemented by a derived media handler if the handler needs to report its idling requirements.

Version Notes
Introduced in QuickTime 6.

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

Declared In
`MediaHandlers.h`

MediaGGetStatus

Reports error conditions to the Movie Toolbox.

```
ComponentResult MediaGGetStatus (
   MediaHandler mh,
   ComponentResult *statusErr
);
```

Parameters

mh

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

statusErr

A pointer to a component result field. If you have error information that you would like to report to the Movie Toolbox, place an appropriate result code into the field referred to by this pointer. See `Error Codes`.

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

Discussion
Your derived media handler should support this function if you anticipate that you may encounter an error when playing your media. Because these errors may include such conditions as low memory or missing hardware, you should only rarely create a derived media handler that does not support this function. If your media handler does not support this function, the base media handler always sets the returned result code to `noErr`.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`MediaHandlers.h`

MediaGSetActiveSegment

Informs your derived media handlers of the current active segment.

```
ComponentResult MediaGSetActiveSegment (
   MediaHandler mh,
   TimeValue activeStart,
   TimeValue activeDuration
);
```

Parameters

mh

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

activeStart

The starting time of the active segment to play. This time value is expressed in your movie's time scale.

activeDuration

A time value that specifies the duration of the active segment. This value is expressed in the movie's time scale.

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

Discussion
Using `SetMovieActiveSegment`, an application can limit the time segment of the movie that will be used for play back. Derived media handlers are given the values for the active segment when this function is called by the Movie Toolbox. Active segment information is usually only needed by media handlers that perform their own scheduling.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`MediaHandlers.h`

MediaGSetIdleManager

Lets a derived media handler report its idling needs.

```
ComponentResult MediaGSetIdleManager (
   MediaHandler mh,
   IdleManager im
);
```

Parameters

mh

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

im

A pointer to an opaque data structure that belongs to the Mac OS Idle Manager. You get this pointer by calling `QTIdleManagerOpen`.

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

Discussion
This routine must be implemented by a derived media handler if the handler needs to retrieve and report its idling requirements.

Version Notes
Introduced in QuickTime 6.

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

Declared In
`MediaHandlers.h`

MediaGSetVolume

Specifies changes to the sound volume setting.

```
ComponentResult MediaGSetVolume (
   MediaHandler mh,
   short volume
);
```

Parameters

mh

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

volume

The media's current volume setting. This value is represented as a 16-bit, fixed-point number. The high-order 8 bits contain the integer portion; 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. The Movie Toolbox scales your media's volume in light of the track's and movie's volume settings, but it does not take into account the system speaker volume setting. This value is appropriate for use with the Sound Manager.

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

Discussion
Your derived media handler should support this function if it can play sounds.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`MediaHandlers.h`

MediaHasCharacteristic

Called by Movie Toolbox with a specified characteristic to allow tracks to be identified by various attributes.

```
ComponentResult MediaHasCharacteristic (
   MediaHandler mh,
   OSType characteristic,
   Boolean *hasIt
);
```

Parameters

mh

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

characteristic

A constant that specifies the attribute of a track. Examples of characteristics that are currently defined are the constants `VisualMediaCharacteristic` and `AudioMediaCharacteristic`.

hasIt

A pointer to a Boolean value that specifies whether the track has the attribute specified in the characteristic parameter. Set this value to TRUE if the attribute applies to your media handler; otherwise, set this value to FALSE.

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

Discussion
You should implement this function for any media handler that has characteristics in addition to spatial ones. If you have set the `handlerHasSpatial` capabilities flag, the base media handler automatically handles the `VisualMediaCharacteristic` constant for you.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
ElectricImageComponent
ElectricImageComponent.win

Declared In
`MediaHandlers.h`

MediaHitTestForTargetRefCon

Locates an object for hit testing.

```
ComponentResult MediaHitTestForTargetRefCon (
   MediaHandler mh,
   long flags,
   Point loc,
   long *targetRefCon
);
```

Parameters

mh

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

flags

Flags (see below) that define the hit. The `mHitTestImage` and `mHitTestInvisible` flags are set by the Standard Controller before this call is made. See these constants:

```
mHitTestBounds
mHitTestImage
mHitTestInvisible
mHitTestIsClick
```

loc

The location of the mouse.

targetRefCon

Returns a reference constant representing an object you're interested in. If this reference constant is not 0, your media handler will receive calls to `MediaGetActionsForQTEvent` (page 19).

Return Value

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.

Declared In

`MediaHandlers.h`

MediaHitTestTargetRefCon

Detects if the mouse click and its release are in the same location and within the object.

```
ComponentResult MediaHitTestTargetRefCon (
   MediaHandler mh,
   long targetRefCon,
   long flags,
   Point loc,
   Boolean *wasHit
);
```

Parameters

mh

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

targetRefCon

A reference constant set by the media handler in a call to `MediaHitTestForTargetRefCon` (page 43).

flags

Flags (see below) that define the hit. See these constants:

```
mHitTestBounds
mHitTestImage
mHitTestInvisible
mHitTestIsClick
```

loc

The location of the mouse.

wasHit

A pointer to a Boolean; it is TRUE if there was a hit, FALSE otherwise.

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

Discussion
This function is called after `MediaGetActionsForQTEvent` (page 19) if a reference constant was set in `MediaHitTestForTargetRefCon` (page 43).

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`MediaHandlers.h`

MediaIdle

Provides processing time to a derived media handler during movie playback.

```
ComponentResult MediaIdle (
   MediaHandler mh,
   TimeValue atMediaTime,
   long flagsIn,
   long *flagsOut,
   const TimeRecord *movieTime
);
```

Parameters

mh

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

atMediaTime

The current time, in your media's time base. You can use this value to obtain the appropriate samples and sample descriptions from your media (using the Movie Toolbox's `GetMediaSample` function). Your media handler may then work with the sample data and descriptions as appropriate.

flagsIn

Contains flags (see below) that indicate what the Movie Toolbox wants your media handler to do. These flags are applicable only to media handlers that perform their own scheduling. The toolbox may use none, or it may set one or more flag to 1. Your handler should examine the `flagsIn` parameter each time the Movie Toolbox calls its `MediaIdle` function. The flags in this parameter indicate the actions that your handler may perform. In addition, when you return from your `MediaIdle` function, you should report what you did using the `flagsOut` parameter. You tell the base media handler that you perform your own scheduling by setting the `handlerNoScheduler` flag to 1 in the `flags` parameter of the `MediaSetHandlerCapabilities` (page 65) function. See these constants:

```
mMustDraw
mAtEnd
mPartialDraw
mPreflightDraw
```

flagsOut

Flags (see below) that are contained in a pointer to a long integer that your media handler uses to indicate to the Movie Toolbox what the handler did. You must always set the values of these flags appropriately. See these constants:

```
mDidDraw
mNeedsToDraw
```

movieTime

A pointer to the movie time value corresponding to the `atMediaTime` parameter. Note that this may differ from the current value returned by `GetMovieTime`.

Return Value

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

Discussion

From time to time, your derived media handler component may determine that only a portion of the available drawing area needs to be redrawn. You can signal that condition to the base media handler component by setting the `mPartialDraw` flag to 1 in the flags your component returns to the Movie Toolbox from your `MediaIdle` function. You return these flags using the `flagsOut` parameter. Whenever you set this flag to 1, the Movie Toolbox calls your component's `MediaGetDrawingRgn` (page 21) function in order to determine the portion of the image that needs to be redrawn.

As an example, consider a full-screen animation. Only rarely is the entire image in motion. Typically, only a small portion of the screen image moves. By using partial redrawing, you can significantly improve the playback performance of such a movie.

Your derived media handler should support this function if you need to do work during movie playback. If you set the `handlerNoIdle` flag to 1 in the `flags` parameter of `MediaSetHandlerCapabilities` (page 65), the Movie Toolbox does not call your `MediaIdle` function. If you encounter an error, save the result code. The Movie Toolbox polls you for status information using `MediaGGetStatus` (page 40).

Special Considerations

If your media handler changes any of the settings of the movie's graphics port or graphics world, be sure to restore the original settings before you exit. In addition, note that you may be drawing into a black-and-white graphics port. Finally, be aware that the Movie Toolbox also uses this function to obtain data for QuickDraw pictures. Therefore, if your media handler does not use QuickDraw when drawing to the screen, be sure to examine the `picSave` field in the graphics port so that you can detect when the Movie Toolbox wants to save an image. Your media handler is then responsible for performing the appropriate display processing.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`MediaHandlers.h`

MediaInitialize

Prepares a derived media handler component to provide access to its media.

```
ComponentResult MediaInitialize (
   MediaHandler mh,
   GetMovieCompleteParams *gmc
);
```

Parameters

mh

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

gmc

A pointer to a `GetMovieCompleteParams` structure. You can obtain information about the current media from this structure. You should copy any values you need to save into your derived media handler's local data area. Because this data structure is owned by the Movie Toolbox, you do not need to worry about disposing of any of the data in it.

Return Value
See `Error Codes`. Returns `noErr` if there is no error. If you return an error, the Movie Toolbox disables the track that uses your media. In cases where your media has just been created, the Movie Toolbox immediately disposes of your media.

Discussion
This function gives your media handler an opportunity to get ready to support the Movie Toolbox. As part of these preparations, your derived media handler should report its capabilities to the base media handler by calling `MediaSetHandlerCapabilities` (page 65). You may choose to examine the data in the `GetMovieCompleteParams` structure; you may also save values from this structure. If you save references to structures (such as the matte pixel map), do not dispose of the memory associated with these structures. The Movie Toolbox owns these structures.

Note that the Movie Toolbox may call other functions supported by your media handler before it calls your `MediaInitialize` function. In particular, it may call your `MediaGetMediaInfo` (page 25) and `MediaPutMediaInfo` (page 53) functions. However, before the Movie Toolbox tries to do anything with the data in your media, it will call your `MediaInitialize` function. The Movie Toolbox loads the movie's data using functions that are supported by the base media handler; your media handler does not have to support those functions.

Special Considerations

All derived media handlers should support this function. In addition, if your media handler saves values from the `GetMovieCompleteParams` structure that may change, be sure to support the corresponding functions that allow the Movie Toolbox to report changes to your media handler. For example, if your handler saves the movie time scale from the `movieScale` field, you should also support the `MediaSetMovieTimeScale` (page 68) function.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
SurfaceVertexProgram

Declared In
`MediaHandlers.h`

MediaInvalidateRegion

Updates the invalidated display region the next time MediaIdle is called.

```
ComponentResult MediaInvalidateRegion (
   MediaHandler mh,
   RgnHandle invalRgn
);
```

Parameters

mh

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

invalRgn

A handle to a region that has been invalidated. Your media handler should not dispose or modify this region. The `invalRgn` parameter is never `NIL`.

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

Discussion
This function is called by the Movie Toolbox when `UpdateMovie` or `InvalidateMovieRegion` is called with a region that intersects your media's track. Derived media handlers need to implement `MediaInvalidateRegion` only if they can perform efficient updates on a portion of their display area.

If a media handler implements this function, it is responsible for ensuring that the appropriate areas of the screen are updated on the next call to `MediaIdle` (page 45). If a media handler does not implement this function, the base media handler sets the `mMustDraw` flag the next time `MediaIdle` is called.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`MediaHandlers.h`

MediaMakeMediaTimeTable

Called by the base media handler to create a media time table.

```
ComponentResult MediaMakeMediaTimeTable (
   MediaHandler mh,
   long **offsets,
   TimeValue startTime,
   TimeValue endTime,
   TimeValue timeIncrement,
   short firstDataRefIndex,
   short lastDataRefIndex,
   long *retDataRefSkew
);
```

Parameters

mh

The media handler to create the time table. You can obtain this reference from `GetMediaHandler`.

offsets

A handle to an unlocked relocatable memory block that is allocated by an application or other software when it calls `QTMovieNeedsTimeTable`, `GetMaxLoadedTimeInMovie`, `MakeTrackTimeTable`, or `MakeMediaTimeTable`. Your derived media handler returns the time table for the media in this block. Your media handler has to resize the handle.

startTime

The first point of the media to be included in the time table. This time value is expressed in the media's time coordinate system.

endTime

The last point of the media to be included in the time table. This time value is expressed in the media's time coordinate system.

timeIncrement

The resolution of the time table. The values in a time table are for a points in the media, and these points are separated by the amount of time specified by this parameter. The time value is expressed in the media's time coordinate system.

firstDataRefIndex

The first in the range of data reference indexes you are querying.

lastDataRefIndex

The last in the range of data reference indexes you are querying.

retDataRefSkew

A pointer to the number of entries, i.e., the number of entries in the offset table per data reference.

Return Value

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

Discussion

The Movie Toolbox calls your derived media handler's `MediaMakeMediaTimeTable` function whenever an application or other software calls the Toolbox's `QTMovieNeedsTimeTable`, `GetMaxLoadedTimeInMovie`, `MakeTrackTimeTable`, or `MakeMediaTimeTable` function. When an application or other software calls one of these functions, it allocates an unlocked relocatable memory block for the time table to be returned and passes a handle to it in the `offsets` parameter. Your derived media handler must resize the block to accommodate the time table it returns.

The time table your derived media handler returns is a two-dimensional array of long integers that is organized so that each row in the table contains values for one data reference. The first column in the table contains values for the time in the media specified by the `startTime` parameter, and each subsequent column

contains values for the point in the media that is later by the value specified by the `timeIncrement` parameter. Each long integer value in the table specifies the offset, in bytes, from the beginning of the data reference for that point in the media.

Special Considerations

The number of columns in the table returned by your derived media handler must be equal to `(endTime - startTime) / timeIncrement`, rounded up. It must also return the offset to the next row of the time table, in long integers, in the `retdataRefSkew` parameter. Because of alignment issues, this value is not always equal to `(endTime - startTime) / timeIncrement` rounded up.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`MediaHandlers.h`

MediaMCIsPlayerEvent

Undocumented

```
ComponentResult MediaMCIsPlayerEvent (
   MediaHandler mh,
   const EventRecord *e,
   Boolean *handledIt
);
```

Parameters

mh

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

e

A pointer to an `EventRecord` structure.

handledIt

A pointer to a Boolean; it is TRUE if the event was handled, FALSE otherwise.

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

Version Notes
Introduced in QuickTime 4.1.

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

Declared In
`MediaHandlers.h`

MediaNavigateTargetRefCon

Locates the object for keyboard focus.

```
ComponentResult MediaNavigateTargetRefCon (
   MediaHandler mh,
   long navigation,
   long *refCon
);
```

Parameters

mh

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

navigation

Flags (see below) that define the direction of navigation. These flags are set by the standard controller, which follows the user's interaction with the tab key, shift key, and mouse. See these constants:

`kRefConNavigationNext`

`kRefConNavigationPrevious`

refCon

Returns a reference constant representing an object you're interested in. If this reference constant is not 0, your media handler will receive calls to `MediaRefConSetProperty` (page 56) and `MediaRefConGetProperty` (page 55).

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
`MediaHandlers.h`

MediaPrePrerollBegin

Undocumented

```
ComponentResult MediaPrePrerollBegin (
   MediaHandler mh,
   TimeValue time,
   Fixed rate,
   PrePrerollCompleteUPP completeProc,
   void *refcon
);
```

Parameters

mh

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

time

Undocumented

rate

Undocumented

completeProc

A PrePrerollCompleteProc callback.

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.

Return Value
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.

Declared In
MediaHandlers.h

MediaPrePrerollCancel

Cancels a media handler pre-preroll operation that was started by MediaPrePrerollBegin.

```
ComponentResult MediaPrePrerollCancel (
   MediaHandler mh,
   void *refcon
);
```

Parameters

mh

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

refcon

The reference constant that was passed to your PrePrerollCompleteProc callback.

Return Value
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.

Declared In
MediaHandlers.h

MediaPreroll

Prepares a media handler for playback.

```
ComponentResult MediaPreroll (
   MediaHandler mh,
   TimeValue time,
   Fixed rate
);
```

Parameters

mh

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

time

The starting time of the media segment to play. This time value is expressed in your movie's time scale.

rate

The rate at which the Movie Toolbox expects to play the media. This is a 32-bit, fixed-point number. Positive values indicate forward rates; negative values correspond to reverse rates.

Return Value
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.

Declared In
`MediaHandlers.h`

MediaPutMediaInfo

Lets a derived media handler store proprietary information in its media.

```
ComponentResult MediaPutMediaInfo (
   MediaHandler mh,
   Handle h
);
```

Parameters

mh

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

h

A handle to storage into which your media handler may place its proprietary information. You determine the format and content of the data that you store in this handle. Your media handler must resize the handle as appropriate before you exit this function. Do not dispose of this handle; it is owned by the Movie Toolbox. The Movie Toolbox uses the base media handler to write this data to your media.

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

Discussion
Whenever the Movie Toolbox opens your media, it provides this private data to your media handler by calling your `MediaGetMediaInfo` (page 25) function. Note that the Movie Toolbox may call this function before it calls your `MediaInitialize` (page 47) function. Your derived media handler should support this function if you need to store private data in your media.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`MediaHandlers.h`

MediaQueueNonPrimarySourceData

Undocumented

```
ComponentResult MediaQueueNonPrimarySourceData (
   MediaHandler mh,
   long inputIndex,
   long dataDescriptionSeed,
   Handle dataDescription,
   void *data,
   long dataSize,
   ICMCompletionProcRecordPtr asyncCompletionProc,
   const ICMFrameTimeRecord *frameTime,
   ICMConvertDataFormatUPP transferProc,
   void *refCon
);
```

Parameters

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

inputIndex
The ID of the entry in the media's input map to which the queued data corresponds.

dataDescriptionSeed
Undocumented

dataDescription
Undocumented

data
Undocumented

dataSize
Undocumented

asyncCompletionProc
A pointer to an `ICMCompletionProcRecord` structure.

frameTime
A pointer to an `ICMFrameTimeRecord` structure.

transferProc
Undocumented

refCon
Undocumented

Return Value
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.

Declared In
`MediaHandlers.h`

MediaRefConGetProperty

Returns the current media handler state based on the property type.

```
ComponentResult MediaRefConGetProperty (
   MediaHandler mh,
   long refCon,
   long propertyType,
   void *propertyValue
);
```

Parameters

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

refCon
The reference constant set by the media handler in `MediaNavigateTargetRefCon` (page 50).

propertyType
The property type sent from the standard controller. Property type values are listed below. See these constants:

`kRefConPropertyCanHaveFocus`
`kRefConPropertyHasFocus`

propertyValue
A pointer to the value that was assigned. Its size is based on the `property` type.

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

Discussion
This routine is called with the reference constant set in `MediaNavigateTargetRefCon` (page 50) to get the current media handler state for a given property type.

Version Notes
Introduced in QuickTime 6.

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

Declared In
`MediaHandlers.h`

MediaRefConSetProperty

Sets a new media handler state based on the property type.

```
ComponentResult MediaRefConSetProperty (
   MediaHandler mh,
   long refCon,
   long propertyType,
   void *propertyValue
);
```

Parameters

mh

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

refCon

The reference constant set by the media handler in `MediaNavigateTargetRefCon` (page 50).

propertyType

The property type sent from the standard controller. Property type values are listed below. See these constants:

`kRefConPropertyCanHaveFocus`

`kRefConPropertyHasFocus`

propertyValue

A pointer to the value to assign. Its size is based on the `property` type.

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

Discussion
This function is called with the reference constant that was set by `MediaNavigateTargetRefCon` (page 50) to set a new media handler state for a given property type.

Version Notes
Introduced in QuickTime 6.

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

Declared In
`MediaHandlers.h`

MediaReleaseSampleDataPointer

Balances calls to MediaGetSampleDataPointer to release allocated memory.

```
ComponentResult MediaReleaseSampleDataPointer (
   MediaHandler mh,
   long sampleNum
);
```

Parameters

mh

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

sampleNum

The number of the sample that is to be released.

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

Discussion
This function should be used only by derived media handlers.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`MediaHandlers.h`

MediaResolveTargetRefCon

Undocumented

```
ComponentResult MediaResolveTargetRefCon (
   MediaHandler mh,
   QTAtomContainer container,
   QTAtom atom,
   long *targetRefCon
);
```

Parameters

mh

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

container

Undocumented

atom

Undocumented

targetRefCon

Undocumented

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

Version Notes
Introduced in QuickTime 4.

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

Declared In
`MediaHandlers.h`

MediaSampleDescriptionB2N

Undocumented

```
ComponentResult MediaSampleDescriptionB2N (
   MediaHandler mh,
   SampleDescriptionHandle sampleDescriptionH
);
```

Parameters

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

sampleDescriptionH
A handle to a `SampleDescription` structure.

Return Value
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.

Declared In
`MediaHandlers.h`

MediaSampleDescriptionChanged

Informs a media handler that SetMediaSampleDescription has been called for a specified sample description.

```
ComponentResult MediaSampleDescriptionChanged (
   MediaHandler mh,
   long index
);
```

Parameters

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

index
The index of the sample description that has been changed.

Return Value
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.

Declared In
`MediaHandlers.h`

MediaSampleDescriptionN2B

Undocumented

```
ComponentResult MediaSampleDescriptionN2B (
   MediaHandler mh,
   SampleDescriptionHandle sampleDescriptionH
);
```

Parameters

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

sampleDescriptionH
Undocumented

Return Value
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.

Declared In
`MediaHandlers.h`

MediaSetActionsCallback

Sets an ActionsProc callback for a media handler.

```
ComponentResult MediaSetActionsCallback (
   MediaHandler mh,
   ActionsUPP actionsCallbackProc,
   void *refcon
);
```

Parameters

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

actionsCallbackProc
A Universal Procedure Pointer to an `ActionsProc` 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 function needs.

Return Value
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.

Declared In
`MediaHandlers.h`

MediaSetActive

Enables and disables media.

```
ComponentResult MediaSetActive (
   MediaHandler mh,
   Boolean enableMedia
);
```

Parameters

mh

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

enableMedia

A Boolean value that indicates whether your media is enabled or disabled. If this parameter is set to TRUE, your media is enabled; if the parameter is FALSE, your media is disabled.

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

Discussion
Your derived media handler should support this function if you perform your own scheduling or if your media handler uses significant amounts of temporary storage. If you are doing your own scheduling (that is, you have set the `handlerNoScheduler` flag to 1 in the `flags` parameter of the `MediaSetHandlerCapabilities` (page 65) function), your media handler needs to keep account of the media's active state so that you can properly respond to Movie Toolbox requests. When your media is disabled, you may choose to dispose of temporary storage you have allocated, so that the storage is available to other programs.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
SurfaceVertexProgram

Declared In
`MediaHandlers.h`

MediaSetChunkManagementFlags

Sets application-global flags that control media chunk management.

```
ComponentResult MediaSetChunkManagementFlags (
   MediaHandler mh,
   UInt32 flags,
   UInt32 flagsMask
);
```

Parameters

mh

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

flags

Constants (see below) that determine chunk management. See these constants:
`kEmptyPurgableChunksOverAllowance`

flagsMask

Undocumented

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

Version Notes
Introduced in QuickTime 6. Can be used only with Mac OS X 10.1 and later.

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

Declared In
`MediaHandlers.h`

MediaSetClip

Specifies changes to a derived media handler's clipping region.

```
ComponentResult MediaSetClip (
   MediaHandler mh,
   RgnHandle theClip
);
```

Parameters

mh

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

theClip

A handle to your media's clipping region. Your media handler is responsible for disposing of this region when you are done with it. Note that this region is defined in the movie's coordinate system.

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

Discussion
Your derived media handler should support this function if you draw during playback. The Movie Toolbox calls this function only if you have set the `handlerHasSpatial` and `handlerCanClip` flags to 1 in the `flags` parameter of the `MediaSetHandlerCapabilities` (page 65) function.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`MediaHandlers.h`

MediaSetDimensions

Informs a media handler when its media's spatial dimensions change.

```
ComponentResult MediaSetDimensions (
   MediaHandler mh,
   Fixed width,
   Fixed height
);
```

Parameters

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

width
The width, in pixels, of the track rectangle. This field, along with the `height` field, specifies a rectangle that surrounds the image that is displayed when the current media is played. This value corresponds to the x coordinate of the lower-right corner of the rectangle and is expressed as a fixed-point number.

height
The height, in pixels, of the track rectangle. This value corresponds to the y coordinate of the lower-right corner of the rectangle and is expressed as a fixed-point number.

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

Discussion
You obtain the initial dimension information from the `width` and height fields of the `GetMovieCompleteParams` structure that the Movie Toolbox provides to your `MediaInitialize` (page 47) function. Your derived media handler should support this function if you draw during playback.

Special Considerations

The Movie Toolbox calls this function only if you have set the `handlerHasSpatial` flag to 1 in the `flags` parameter of the `MediaSetHandlerCapabilities` (page 65) function.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
MediaHandlers.h

MediaSetDoMCActionCallback

Sets a DoMCActionProc callback for a media handler.

```
ComponentResult MediaSetDoMCActionCallback (
   MediaHandler mh,
   DoMCActionUPP doMCActionCallbackProc,
   void *refcon
);
```

Parameters

mh

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

doMCActionCallbackProc

A Universal Procedure Pointer to 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
See Error Codes. Returns noErr if there is no error.

Version Notes
Introduced in QuickTime 4.

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

Declared In
MediaHandlers.h

MediaSetGraphicsMode

Sets the graphics mode and blend color of any media handler.

```
ComponentResult MediaSetGraphicsMode (
   MediaHandler mh,
   long mode,
   const RGBColor *opColor
);
```

Parameters

mh

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

mode

The graphics mode of the media handler; see Graphics Transfer Modes.

opColor

A pointer to the color for use in blending and transparent operations. The media handler passes this color to QuickDraw as appropriate when you draw in `addPin`, `subPin`, `blend`, `transparent`, or `graphicsModeStraightAlphaBlend` mode.

Return Value
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
qtactiontargets
qtactiontargets.win
vrmovies
vrmovies.win
vrscript

Declared In
`MediaHandlers.h`

MediaSetGWorld

Lets a derived media handler learn about changes to its media's graphic environment.

```
ComponentResult MediaSetGWorld (
   MediaHandler mh,
   CGrafPtr aPort,
   GDHandle aGD
);
```

Parameters

mh

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

aPort

A pointer to the new graphics port. Note that this may be either a color or a black-and-white port.

aGD

A handle to the new graphics device.

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

Discussion
Your derived media handler should support this function if you perform specialized graphics processing or if you are using the Image Compression Manager to decompress your media. Note that when the Movie Toolbox calls your `MediaIdle` (page 45) function, it supplies you with information about the current graphics environment. Consequently, you do not need to support the `MediaSetGWorld` function in order to draw

during playback. However, if your media data is compressed and you are using the Image Compression Manager to decompress sequences, you may need to provide updated graphics environment information before playback.

You obtain the initial graphics environment information from the `moviePort` and `movieGD` fields of the `GetMovieCompleteParams` structure that the Movie Toolbox provides to your `MediaInitialize` (page 47) function. The Movie Toolbox calls this function only if you have set the `handlerHasSpatial` flag to 1 in the `flags` parameter of the `MediaSetHandlerCapabilities` (page 65) function.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`MediaHandlers.h`

MediaSetHandlerCapabilities

Lets a derived media handler report its capabilities to the base media handler.

```
ComponentResult MediaSetHandlerCapabilities (
   MediaHandler mh,
   long flags,
   long flagsMask
);
```

Parameters

mh

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

flags

Flags (see below) that specify the capabilities of your derived media handler. This parameter contains a number of flags, each of which corresponds to a particular feature. You may work with more than one flag at a time. Be sure to set unused flags to 0. See these constants:

`handlerHasSpatial`
`handlerCanClip`
`handlerCanMatte`
`handlerCanTransferMode`
`handlerNeedsBuffer`
`handlerNoIdle`
`handlerNoScheduler`
`handlerWantsTime`
`handlerCGrafPortOnly`

flagsMask

Indicates which flags in the `flags` parameter are to be considered in this operation. For each bit in the `flags` parameter that you want the base media handler to consider, you must set the corresponding bit in the `flagsMask` parameter to 1. Set unused flags to 0. This allows you to work with a single flag without altering the settings of other flags.

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

Discussion
Your media handler may call this function at any time. In general, you should call it from your `MediaInitialize` (page 47) function , so that you report your capabilities to the base media handler before the Movie Toolbox starts working with your media. You may call this function again later, in response to changing conditions. For example, if your media handler receives a matrix that it cannot accommodate from the `MediaSetMatrix` (page 67) function, you can allow the base media handler to handle your drawing by calling this function and setting the `handlerNeedsBuffer` flag in both the `flags` parameter and the `flagsMask` parameter to 1.

Special Considerations

Note that this function is provided by the base media handler; your media handler does not support this function.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
AlwaysPreview
qdmediahandler
qdmediahandler.win

Declared In
`MediaHandlers.h`

MediaSetHints

Implements the appropriate behavior for the various media hints such as scrub mode and high-quality mode.

```
ComponentResult MediaSetHints (
   MediaHandler mh,
   long hints
);
```

Parameters

mh

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

hints

All hint bits that currently apply to the given media.

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

Discussion
When an application calls `SetMoviePlayHints` or `SetMediaPlayHints`, your media handler's `MediaSetHints` routine is called.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`MediaHandlers.h`

MediaSetMatrix

Tells a media handler about changes to either the movie matrix or the track matrix.

```
ComponentResult MediaSetMatrix (
   MediaHandler mh,
   MatrixRecord *trackMovieMatrix
);
```

Parameters

mh

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

trackMovieMatrix

A pointer to the matrix that transforms your media's pixels into the movie's coordinate system. The Movie Toolbox obtains this matrix by concatenating the track matrix and the movie matrix. You should use this matrix whenever you are displaying graphical data from your media.

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

Discussion
You obtain the initial matrix from the `trackMovieMatrix` field of the `GetMovieCompleteParams` structure that the Movie Toolbox provides to your `MediaInitialize` (page 47) function. Your derived media handler should support this function if you draw during playback.

Special Considerations

The Movie Toolbox calls this function only if you have set the `handlerHasSpatial` flag to 1 in the `flags` parameter of the `MediaSetHandlerCapabilities` (page 65) function.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`MediaHandlers.h`

MediaSetMediaTimeScale

Informs a media handler that its media's time scale has been changed.

```
ComponentResult MediaSetMediaTimeScale (
   MediaHandler mh,
   TimeScale newTimeScale
);
```

Parameters

mh

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

newTimeScale

Specifies your media's new time scale.

Return Value

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

Discussion

You obtain the initial media time scale information from the `mediaScale` field of the `GetMovieCompleteParams` structure that the Movie Toolbox provides to your `MediaInitialize` (page 47) function.

Special Considerations

Your derived media handler should support this function if your media handler stores time information that pertains to its media.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`MediaHandlers.h`

MediaSetMovieTimeScale

Informs a media handler that the movie's time scale has been changed.

```
ComponentResult MediaSetMovieTimeScale (
   MediaHandler mh,
   TimeScale newTimeScale
);
```

Parameters

mh

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

newTimeScale

The movie's new time scale.

Return Value

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

Discussion
You obtain the initial movie time scale information from the `movieScale` field of the `GetMovieCompleteParams` structure that the Movie Toolbox provides to your `MediaInitialize` (page 47) function.

Special Considerations

Your derived media handler should support this function if your media handler stores time information in the movie's time coordinate system.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`MediaHandlers.h`

MediaSetNonPrimarySourceData

Allows a media handler to support receiving media data from other media handlers.

```
ComponentResult MediaSetNonPrimarySourceData (
   MediaHandler mh,
   long inputIndex,
   long dataDescriptionSeed,
   Handle dataDescription,
   void *data,
   long dataSize,
   ICMCompletionProcRecordPtr asyncCompletionProc,
   ICMConvertDataFormatUPP transferProc,
   void *refCon
);
```

Parameters

mh

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

inputIndex

This value is the ID of the entry in the media's input map to which the data provided by the call corresponds.

dataDescriptionSeed

This value is changed each time the `dataDescription` has changed. This allows for a quick check by the media handler to see if the `dataDescription` has changed.

dataDescription

A handle to a data structure describing the input data.

data

A pointer to the input data. This pointer must contain a 32-bit address.

dataSize

The size of the sample in bytes.

asyncCompletionProc

A pointer to a `ICMCompletionProcRecord` structure. If `asyncCompletionProc` is set to `NIL`, the data pointer will be valid only for the duration of this call. If `asyncCompletionProc` is not `NIL`, it contains an `ICMCompletionProcRecord` structure that must be called when your media handler is done with the provided data pointer.

transferProc

A routine that allows the application to transform the type of the input data to the kind of data preferred by the codec. The client of the codec passes the source data in the form most convenient for it. If the codec needs the data in another form, it can negotiate with the caller or directly with the Image Compression Manager to obtain the required data format.

refCon

A reference constant, defined as a void pointer. Your application specifies the value of this reference constant in the function structure you pass to the media handler.

Return Value

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

Discussion

There are two tasks required to support modifier tracks in a derived media handler: sending and receiving. The base media handler takes care of sending data for all its clients. Therefore, authors of derived media handlers do not usually need to implement sending data support.

Receiving data is a more complex situation. The base media handler takes care of input types that it understands. The base media handler supports the following types of data: If a media handler wants to support receiving other types of data it must implement `MediaSetNonPrimarySourceData`. `MediaSetNonPrimarySourceData` is called by modified tracks to supply the current data for each input. All unrecognized input types should be delegated to the base media handler so that they can be handled.

The following is a basic shell implementation of a derived media handler's `MediaSetNonPrimarySourceData` function. Note that your derived media handler must delegate all input types it does not handle to the base media handler:

```
// MySetNonPrimarySourceData coding example
kTrackModifierTypeMatrix
kTrackModifierTypeGraphicsMode
kTrackModifierTypeClip
kTrackModifierTypeVolume
kTrackModifierTypeBalance
pascal ComponentResult MySetNonPrimarySourceData( MyGlobals store,
    long inputIndex, long dataDescriptionSeed, Handle dataDescription,
    void *data, long dataSize,
    ICMCompletionProcRecordPtr asyncCompletionProc,
    UniversalProcPtr transferProc, void *refCon )
{
ComponentResult err =noErr;
QTAtom inputAtom;
QTAtom typesAtom;
long inputType;
// determine what kind of input this is
inputAtom =QTFindChildByID(store->
inputMap,
    kParentAtomIsContainer, kTrackModifierInput, inputIndex, NIL);
if (!inputAtom) {
    err =cannotFindAtomErr;
    goto bail;
}
```

```
    typesAtom =QTFindChildByID(store->
inputMap, inputAtom,
    kTrackModifierType, 1, NIL);
err =QTCopyAtomDataToPtr(store->
inputMap, typesAtom, FALSE,
    sizeof(inputType), &inputType, NIL);
if (err) goto bail;
switch(inputType) {
    case kMyInputType:
        if (data ==NIL) {
            // no data, reset to default value
        }
        else {
            // use this data
            // when done, notify caller we're done with this data
            if (asyncCompletionProc)
                CallICMCompletionProc(
                    asyncCompletionProc->
completionProc,
                    noErr, codecCompletionSource | codecCompletionDest,
                    asyncCompletionProc->
completionRefCon);
        }
        break;
```

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`MediaHandlers.h`

MediaSetPublicInfo

Undocumented

```
ComponentResult MediaSetPublicInfo (
   MediaHandler mh,
   OSType infoSelector,
   void *infoDataPtr,
   Size dataSize
);
```

Parameters

mh

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

infoSelector

Undocumented

infoDataPtr

Undocumented

dataSize

Undocumented

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

Version Notes
Introduced in QuickTime 5.

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

Related Sample Code
qtskins
qtskins.win

Declared In
`MediaHandlers.h`

MediaSetPurgeableChunkMemoryAllowance

Sets the maximum amount of memory that QuickTime will allow purgeable chunks to occupy.

```
ComponentResult MediaSetPurgeableChunkMemoryAllowance (
   MediaHandler mh,
   Size allowance
);
```

Parameters

mh

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

allowance

The number of bytes allowed.

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

Discussion
This is an application-global setting.

Version Notes
Introduced in QuickTime 6. Can be used only with Mac OS X 10.1 and later.

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

Declared In
`MediaHandlers.h`

MediaSetRate

Sets a media's playback rate.

```
ComponentResult MediaSetRate (
   MediaHandler mh,
   Fixed rate
);
```

Parameters

mh

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

rate

A 32-bit, fixed-point number that indicates your media's new effective playback rate. This effective rate accounts for any master time bases that may be in use with the current movie. Positive values represent forward rates and negative values indicate reverse rates.

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

Discussion
You obtain the initial rate information from the `effectiveRate` field of the `GetMovieCompleteParams` structure that the Movie Toolbox provides to your `MediaInitialize` (page 47) function. Your derived media handler should support this function if you perform your own scheduling; that is, you have set the `handlerNoScheduler` flag to 1 in the `flags` parameter of `MediaSetHandlerCapabilities` (page 65). Your media handler can use this function to determine when your media is playing, and the direction and rate of playback. This information can help you prepare for playback more efficiently.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`MediaHandlers.h`

MediaSetScreenLock

Locks the display screen for a media handler.

```
ComponentResult MediaSetScreenLock (
   MediaHandler mh,
   Boolean lockIt
);
```

Parameters

mh

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

lockIt

Pass TRUE to lock the screen, FALSE to unlock it.

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

Version Notes
Introduced in QuickTime 4.

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

Declared In
`MediaHandlers.h`

MediaSetSoundBalance

Sets the right/left sound balance of a track.

```
ComponentResult MediaSetSoundBalance (
   MediaHandler mh,
   short balance
);
```

Parameters

mh

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

balance

The new balance setting for the media handler's track. The balance setting is a signed 16-bit integer that controls the relative volume of the left and right sound channels. A value of 0 sets the balance to neutral. Positive values shift the balance to the right channel, negative values to the left channel. The valid range is 127 (right channel only) to -128 (left channel only).

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

Discussion
Use this function to alter the audio balance on a per-track basis. It works with both mono and stereo sources. The actual volume of the audio is not changed by this setting, only its relative distribution. This function operates on sound tracks, music tracks, and Flash tracks containing audio. It may operate on other audio track types as well; if the the chosen media handler does not support this function, it returns `badComponentSelector`. To obtain the media handler for a track, call `GetTrackMedia` and then `GetMediaHandler`.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
vrscript
vrscript.win

Declared In
`MediaHandlers.h`

MediaSetSoundBassAndTreble

Sets the bass and treble controls for a media handler.

```
ComponentResult MediaSetSoundBassAndTreble (
   MediaHandler mh,
   short bass,
   short treble
);
```

Parameters

mh

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

bass

Undocumented

treble

Undocumented

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

Version Notes
Introduced in QuickTime 4.

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

Declared In
`MediaHandlers.h`

MediaSetSoundEqualizerBands

Sets sound equalizer bands for a media handler.

```
ComponentResult MediaSetSoundEqualizerBands (
   MediaHandler mh,
   MediaEQSpectrumBandsRecordPtr spectrumInfo
);
```

Parameters

mh

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

spectrumInfo

A pointer to a `MediaEQSpectrumBandsRecord` structure.

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

Version Notes
Introduced in QuickTime 4.

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

Related Sample Code
sndequalizer

SurfaceVertexProgram

Declared In
MediaHandlers.h

MediaSetSoundLevelMeteringEnabled

Enables or disables sound level metering for a media handler.

```
ComponentResult MediaSetSoundLevelMeteringEnabled (
   MediaHandler mh,
   Boolean enable
);
```

Parameters

mh

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

enable

Pass TRUE to enable sound level metering, FALSE to disable it.

Return Value
See Error Codes. Returns noErr if there is no error.

Version Notes
Introduced in QuickTime 4.

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

Related Sample Code
SurfaceVertexProgram

Declared In
MediaHandlers.h

MediaSetSoundLocalizationData

Supports 3D sound capabilities in a media handler that plays sound.

```
ComponentResult MediaSetSoundLocalizationData (
   MediaHandler mh,
   Handle data
);
```

Parameters

mh

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

data

The data passed to your media handler, in the format of a Macintosh Sound Sprockets SSpLocalizationData structure.

Return Value
See Error Codes. Returns noErr if there is no error.

Discussion
This routine is passed a handle containing the new `SSpLocalizationData` structure to use. If the handle is `NIL`, it indicates that no 3D sound effects should be used. If you implement this routine, and return `noErr` as the result, it is assumed that your media handle assumes responsibility for disposing of the data handle passed. If the implementation of this routine returns an error, the caller will dispose of the handle. The reason for this behavior is to minimize the copying of the settings handle, making it easier for developers to implement this function. This function is called regardless of whether the 3D sound settings were set on the track using `SetTrackSoundLocalizationSettings` or via a modifier track mechanism.

Special Considerations

If you are creating a media handler that plays sound and wish to support 3D sound capabilities, you need to implement this routine.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`MediaHandlers.h`

MediaSetSoundOutputComponent

Sets the sound output component for a media handler.

```
ComponentResult MediaSetSoundOutputComponent (
   MediaHandler mh,
   Component outputComponent
);
```

Parameters

mh

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

outputComponent

An instance of a sound output component. Your software obtains this reference when calling `OpenComponent` or `OpenDefaultComponent`.

Return Value
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.

Declared In
`MediaHandlers.h`

MediaSetTrackInputMapReference

Provides a derived media handler with an updated input map.

```
ComponentResult MediaSetTrackInputMapReference (
   MediaHandler mh,
   QTAtomContainer inputMap
);
```

Parameters

mh

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

inputMap

The media input map for this operation. Do not modify or dispose of the input map provided.

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

Discussion
When an application modifies the media input map, this function provides the derived media handler with the updated input map. When this function is called, the media handler should store the updated input map and recheck the types of all inputs, if it is caching this information. The input map reference passed to this function should not be disposed of or modified by the media handler.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`MediaHandlers.h`

MediaSetUserPreferredCodecs

Requests that a media handler favor specified codec components when selecting components with which to play media.

```
ComponentResult MediaSetUserPreferredCodecs (
   MediaHandler mh,
   CodecComponentHandle userPreferredCodecs
);
```

Parameters

mh

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

userPreferredCodecs

A handle containing component identifiers. The media handler component will make its own copy of this handle. The components should be specified in order from most preferred to least preferred. Pass `NIL` to invalidate the standing request without substituting another.

Return Value
See `Error Codes`. Returns `badComponentSelector` if the media handler component does not support this call. Returns `noErr` if there is no error.

Discussion
This method does not guarantee that the specified components will be used; other factors may take precedence. Components that are preferred may not be used if they can't be part of the chain required to play the media; for example, if they don't handle the pixel format or the video output.

Here is an example of code that uses this function:

```
CodecComponentHandle userPreferredCodecs =nil;
ComponentDescription cd ={ decompressorComponentType,
                           myPreferredCodecType,
                           myPreferredCodecManufacturer,
                           0,
                           0 };
CodecComponent    c =FindNextComponent( 0, &cd );
MediaHandler      myMedia;
OSErr             err;
PtrToHand( &c, (Handle*)&userPreferredCodecs, sizeof(c) );
myMedia =GetMediaHandler( GetTrackMedia( track ) );
err =MediaSetUserPreferredCodecs( myMedia, userPreferredCodecs );
DisposeHandle( (Handle)userPreferredCodecs );
```

Version Notes
Introduced in QuickTime 5.

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

Declared In
`MediaHandlers.h`

MediaSetVideoParam

Lets you dynamically adjust the brightness, contrast, hue, sharpness, saturation, black level, and white level of a video image.

```
ComponentResult MediaSetVideoParam (
   MediaHandler mh,
   long whichParam,
   unsigned short *value
);
```

Parameters

mh

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

whichParam

A constant (see below) that specifies the video parameter you want to adjust. See these constants:

```
kMediaVideoParamBrightness
kMediaVideoParamContrast
kMediaVideoParamHue
kMediaVideoParamSharpness
kMediaVideoParamSaturation
kMediaVideoParamBlackLevel
kMediaVideoParamWhiteLevel
```

value

The actual value of the video parameter. The meaning of the values vary depending on the implementation.

Return Value
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.

Declared In
`MediaHandlers.h`

MediaTargetRefConsEqual

Undocumented

```
ComponentResult MediaTargetRefConsEqual (
   MediaHandler mh,
   long firstRefCon,
   long secondRefCon,
   Boolean *equal
);
```

Parameters

mh

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

firstRefCon

Undocumented

secondRefCon

Undocumented

equal

A pointer to a Boolean; it is TRUE if `firstRefCon` and `secondRefCon` are equal, FALSE otherwise.

Return Value
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.

Declared In
`MediaHandlers.h`

MediaTimeBaseChanged

Undocumented

```
ComponentResult MediaTimeBaseChanged (
   MediaHandler mh
);
```

Parameters

mh

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

Return Value
Undocumented

Version Notes
Introduced in QuickTime 4.

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

Declared In
`MediaHandlers.h`

MediaTrackEdited

Informs a derived media handler about edits to its track.

```
ComponentResult MediaTrackEdited (
   MediaHandler mh
);
```

Parameters

mh

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

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

Discussion
Your derived media handler should support this function if you are caching location information about track edits, or if you are using any time values in the movie's time base. Whenever the Movie Toolbox calls this function, your media handler should recalculate this type of information.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`MediaHandlers.h`

MediaTrackPropertyAtomChanged

Notifies the derived media handler whenever its media property atom has changed.

```
ComponentResult MediaTrackPropertyAtomChanged (
   MediaHandler mh
);
```

Parameters

mh

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

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

Discussion
`MediaTrackPropertyAtomChanged` is called whenever `SetMediaPropertyAtom` is called. If the media handler uses information from the property atom, it should rebuild the information at this time.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`MediaHandlers.h`

MediaTrackReferencesChanged

Notifies the derived media handler whenever the track references in the movie change.

```
ComponentResult MediaTrackReferencesChanged (
   MediaHandler mh
);
```

Parameters

mh

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

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

Discussion
When an application creates, modifies, or deletes a track reference, the media handler's `MediaTrackReferencesChanged` function is called. When this function is called, a media handler should rebuild all information about track references and reset its values for all media inputs to their default values.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`MediaHandlers.h`

MediaVideoOutputChanged

Undocumented

```
ComponentResult MediaVideoOutputChanged (
   MediaHandler mh,
   ComponentInstance vout
);
```

Parameters

mh

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

vout

Undocumented

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

Version Notes
Introduced in QuickTime 5.

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

Declared In
`MediaHandlers.h`

NewPrePrerollCompleteUPP

Allocates a Universal Procedure Pointer for the PrePrerollCompleteProc callback.

```
PrePrerollCompleteUPP NewPrePrerollCompleteUPP (
   PrePrerollCompleteProcPtr userRoutine
);
```

Parameters

userRoutine

A pointer to your application-defined function.

Return Value
A new UPP; see `Universal Procedure Pointers`.

Discussion
This function is used with Macintosh PowerPC systems. See *Inside Macintosh: PowerPC System Software*.

Version Notes
Introduced in QuickTime 4.1. Replaces `NewPrePrerollCompleteProc`.

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

Declared In
`MediaHandlers.h`

Callbacks

PrePrerollCompleteProc

Undocumented

```
typedef void (*PrePrerollCompleteProcPtr) (MediaHandler mh, OSErr err, void *refcon);
```

If you name your function `MyPrePrerollCompleteProc`, you would declare it this way:

```
void MyPrePrerollCompleteProc (
    MediaHandler    mh,
    OSErr           err,
    void            *refcon );
```

Parameters

mh
A media handler.

err
An error code; see `Error Codes`.

refcon
Pointer to a reference constant that the client code supplies to your callback. You can use this reference to point to a data structure containing any information your callback needs.

Declared In
`MediaHandlers.h`

Data Types

GetMovieCompleteParams

Defines the layout of the complete movie parameter structure used by MediaInitialize.

```
struct GetMovieCompleteParams {
     short              version;
     Movie              theMovie;
     Track              theTrack;
     Media              theMedia;
     TimeScale          movieScale;
     TimeScale          mediaScale;
     TimeValue          movieDuration;
     TimeValue          trackDuration;
     TimeValue          mediaDuration;
     Fixed              effectiveRate;
     TimeBase           timeBase;
     short              volume;
     Fixed              width;
     Fixed              height;
     MatrixRecord       trackMovieMatrix;
     CGrafPtr           moviePort;
     GDHandle           movieGD;
     PixMapHandle       trackMatte;
     QTAtomContainer    inputMap;
 };
```

Fields

`version`

Discussion
Specifies the version of this structure. This field is always set to 1.

`theMovie`

Discussion
Identifies the movie that contains the current media's track. This movie identifier is supplied by the Movie Toolbox. Your component may use this identifier to obtain information about the movie that is using your media.

`theTrack`

Discussion
Identifies the track that contains the current media. This track identifier is supplied by the Movie Toolbox. Your component may use this identifier to obtain information about the track that contains your media. For example, you might call `GetTrackNextInterestingTime` to examine the track's edit list.

`theMedia`

Discussion
Identifies the current media. This media identifier is supplied by the Movie Toolbox. Your derived media handler can use this identifier to read samples or sample descriptions from the current media, using `GetMediaSample` and `GetMediaSampleDescription`.

`movieScale`

Discussion
Specifies the time scale of the movie that contains the current media's track. If the Movie Toolbox changes the movie's time scale, the toolbox calls your derived media handler's `MediaSetMovieTimeScale` (page 68) function.

`mediaScale`

Discussion
Specifies the time scale of the current media. If the Movie Toolbox changes your media's time scale, the toolbox calls your derived media handler's `MediaSetMediaTimeScale` (page 67) function.

`movieDuration`

Discussion
Contains the movie's duration. This value is expressed in the movie's time scale.

`trackDuration`

Discussion
Contains the track's duration. This value is expressed in the movie's time scale.

`mediaDuration`

Discussion
Contains the media's duration. This value is expressed in the media's time scale.

`effectiveRate`

Discussion
Contains the media's effective rate. This rate ties the media's time scale to the passage of absolute time, and does not necessarily correspond to the movie's rate. This value takes into account any master time bases that may be serving the media's time base. The value of this field indicates the number of time units (in the media's time scale) that pass each second. This rate is represented as a 32-bit, fixed-point number. The high-order 16 bits contain the integer portion, and the low-order 16 bits contain the fractional portion. The rate is negative when time is moving backward for the media. Whenever the Movie Toolbox changes your media's effective rate, it calls your derived media handler's `MediaSetRate` (page 72) function.

`timeBase`

Discussion
Identifies the media's time base.

`volume`

Discussion
Contains the media's current volume setting. This value is represented as a 16-bit, fixed-point number. The high-order 8 bits contain the integer portion; 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. If QuickTime changes your media's volume, it calls your derived media handler's `MediaGSetVolume` (page 42) function.

`width`

Discussion
Indicates the width, in pixels, of the track rectangle. This field, along with the `height` field, specifies a rectangle that surrounds the image that is displayed when the current media is played. This value corresponds to the x coordinate of the lower-right corner of the rectangle and is expressed as a fixed-point number. If the Movie Toolbox modifies this rectangle, the toolbox calls your derived media handler's `MediaSetDimensions` (page 62) function. Note that your media need not present only a rectangular image. The Movie Toolbox can use a clipping region to cause your media's image to be displayed in a region of arbitrary shape, and it can use a matte to control the image's transparency. The toolbox calls your derived media handler's `MediaSetClip` (page 61) function whenever it changes your media's clipping region. The `trackMatte` field in this structure specifies a matte region.

`height`

Discussion
Indicates the height, in pixels, of the track rectangle. This value corresponds to the y coordinate of the lower-right corner of the rectangle and is expressed as a fixed-point number.

`trackMovieMatrix`

Discussion
Specifies the matrix that transforms your media's pixels into the movie's coordinate system. The Movie Toolbox obtains this matrix by concatenating the track matrix and the movie matrix. You should use this matrix whenever you are displaying graphical data from your media. Whenever the Movie Toolbox modifies this matrix, it calls your derived media handler's `MediaSetMatrix` (page 67) function.

`moviePort`

Discussion
Indicates the movie's graphics port. Whenever the Movie Toolbox changes the movie's graphics world, it calls your derived media handler's `MediaSetGWorld` (page 64) function.

`movieGD`

Discussion
Specifies the movie's graphics device. Whenever the Movie Toolbox changes the movie's graphics world, it calls your derived media handler's `MediaSetGWorld` (page 64) function.

`trackMatte`

Discussion
Identifies the matte region assigned to the track that uses your media. This field contains a handle to a pixel map that contains a blend matte. Your component is not responsible for disposing of this matte. If there is no matte, this field is set to `NIL`.

`inputMap`

Discussion
A reference to the media's input map. The media input map should not be modified or disposed.

Related Functions
`MediaInitialize` (page 47)

Declared In
`MediaHandlers.h`

LevelMeterInfo

Contains sound level meter readings.

```
struct LevelMeterInfo {
     short    numChannels;
     UInt8    leftMeter;
     UInt8    rightMeter;
 };
```

Fields
`numChannels`

Discussion
Contains 1 for mono or 2 for stereo source.

`leftMeter`

Discussion
Left meter level, 0-255 range.

`rightMeter`

Discussion
Right meter level, 0-255 range.

Related Functions
`MediaGetSoundLevelMeterInfo` (page 34)

Declared In
`MediaHandlers.h`

LevelMeterInfoPtr

Represents a type used by the Media Handler API.

```
typedef LevelMeterInfo * LevelMeterInfoPtr;
```

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

Declared In
`Sound.h`

MediaEQSpectrumBandsRecord

Provides data for MediaGetSoundEqualizerBands and MediaSetSoundEqualizerBands.

```
struct MediaEQSpectrumBandsRecord {
     short               count;
     UnsignedFixedPtr    frequency;
 };
```

Fields
`count`

Discussion
Number of frequencies in this structure.

`frequency`

Discussion
Pointer to array of frequencies.

Related Functions
`MediaGetSoundEqualizerBands` (page 33)
`MediaSetSoundEqualizerBands` (page 75)

Declared In
`MediaHandlers.h`

MediaEQSpectrumBandsRecordPtr

Represents a type used by the Media Handler API.

```
typedef MediaEQSpectrumBandsRecord * MediaEQSpectrumBandsRecordPtr;
```

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

Declared In
`MediaHandlers.h`

PrePrerollCompleteUPP

Represents a type used by the Media Handler API.

```
typedef STACK_UPP_TYPE(PrePrerollCompleteProcPtr) PrePrerollCompleteUPP;
```

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

Declared In
`MediaHandlers.h`

QTCustomActionTargetPtr

Represents a type used by the Media Handler API.

```
typedef QTCustomActionTargetRecord * QTCustomActionTargetPtr;
```

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

Declared In
`MediaHandlers.h`

QTCustomActionTargetRecord

Defines a target for CallComponentExecuteWiredAction.

```
struct QTCustomActionTargetRecord {
     Movie            movie;
     DoMCActionUPP    doMCActionCallbackProc;
     long             callBackRefcon;
     Track            track;
     long             trackObjectRefCon;
     Track            defaultTrack;
     long             defaultObjectRefCon;
     long             reserved1;
     long             reserved2;
 };
```

Fields
`movie`

Discussion
A movie identifier obtained from such functions as `NewMovie`, `NewMovieFromFile`, and `NewMovieFromHandle`.

`doMCActionCallbackProc`

Discussion
A Universal Procedure Pointer that accesses a `DoMCActionProc` callback.

`callBackRefcon`

Discussion
A reference constant to be passed to the `DoMCActionProc` callback. Use this value to point to a data structure containing any information the callback needs.

`track`

Discussion
A track identifier obtained from such functions as `NewMovieTrack` and `GetMovieTrack`.

`trackObjectRefCon`

Discussion
Undocumented

`defaultTrack`

Discussion
The identifier of a default track, obtained from such functions as `NewMovieTrack` and `GetMovieTrack`.

`defaultObjectRefCon`

Discussion
Undocumented

`reserved1`

Discussion
Reserved.

`reserved2`

Discussion
Reserved.

Declared In
`MediaHandlers.h`

Constants

MediaForceUpdate Values

Constants passed to MediaForceUpdate.

```
enum {
  forceUpdateRedraw             = 1 << 0,
  forceUpdateNewBuffer          = 1 << 1
};
```

Declared In
`MediaHandlers.h`

Data Handler Flags

Constants that represent data handler flags.

```
enum {
  handlerHasSpatial             = 1 << 0,
  handlerCanClip                = 1 << 1,
  handlerCanMatte               = 1 << 2,
  handlerCanTransferMode        = 1 << 3,
  handlerNeedsBuffer            = 1 << 4,
  handlerNoIdle                 = 1 << 5,
  handlerNoScheduler            = 1 << 6,
  handlerWantsTime              = 1 << 7,
  handlerCGrafPortOnly          = 1 << 8,
  handlerCanSend                = 1 << 9,
  handlerCanHandleComplexMatrix = 1 << 10,
  handlerWantsDestinationPixels = 1 << 11,
  handlerCanSendImageData       = 1 << 12,
  handlerCanPicSave             = 1 << 13
};
```

Declared In
MediaHandlers.h

MediaSetChunkManagementFlags Values

Constants passed to MediaSetChunkManagementFlags.

```
enum {
  kEmptyPurgableChunksOverAllowance = 1
};
```

Declared In
MediaHandlers.h

MediaSetVideoParam Values

Constants passed to MediaSetVideoParam.

```
enum {
  kMediaVideoParamBrightness    = 1,
  kMediaVideoParamContrast      = 2,
  kMediaVideoParamHue           = 3,
  kMediaVideoParamSharpness     = 4,
  kMediaVideoParamSaturation    = 5,
  kMediaVideoParamBlackLevel    = 6,
  kMediaVideoParamWhiteLevel    = 7
};
```

Declared In
MediaHandlers.h

MediaNavigateTargetRefCon Values

Constants passed to MediaNavigateTargetRefCon.

```
enum {
  kRefConNavigationNext         = 0,
  kRefConNavigationPrevious     = 1
};
```

Declared In
MediaHandlers.h

MediaRefConSetProperty Values

Constants passed to MediaRefConSetProperty.

```
enum {
  kRefConPropertyCanHaveFocus   = 1,    /* Boolean */
  kRefConPropertyHasFocus       = 2     /* Boolean */
};
```

Declared In
MediaHandlers.h

Media Task Flags

Constants that represent flags for media tasks.

```
enum {
  mDidDraw                      = 1 << 0,
  mNeedsToDraw                  = 1 << 2,
  mDrawAgain                    = 1 << 3,
  mPartialDraw                  = 1 << 4,
  mWantIdleActions              = 1 << 5
};
enum {
  mMustDraw                     = 1 << 3,
  mAtEnd                        = 1 << 4,
  mPreflightDraw                = 1 << 5,
  mSyncDrawing                  = 1 << 6,
  mPrecompositeOnly             = 1 << 9,
  mSoundOnly                    = 1 << 10,
  mDoIdleActionsBeforeDraws     = 1 << 11,
  mDisableIdleActions           = 1 << 12
};
enum {
  mOpaque                       = 1L << 0,
  mInvisible                    = 1L << 1
};
```

Declared In
MediaHandlers.h

MediaHitTestTargetRefCon Values

Constants passed to MediaHitTestTargetRefCon.

```
enum {
  mHitTestBounds                = 1L << 0, /*    point must only be within
targetRefCon's bounding box */
  mHitTestImage                 = 1L << 1, /*  point must be within the shape of
the targetRefCon's image */
  mHitTestInvisible             = 1L << 2, /*  invisible targetRefCon's may be hit
 tested */
  mHitTestIsClick               = 1L << 3 /*  for codecs that want mouse events */
};
```

Declared In

`MediaHandlers.h`

Document Revision History

This table describes the changes to *Media Types and Media Handlers Reference*.

Date	Notes
2006-11-10	Correct minor typos.
2006-05-23	New document, based on previously published material, that describes the API for QuickTime media handlers.

Document Revision History

Index

C

CallComponentExecuteWiredAction function 13

D

Data Handler Flags 91
DisposePrePrerollCompleteUPP function 13

G

GetMovieCompleteParams structure 84

L

LevelMeterInfo structure 87
LevelMeterInfoPtr data type 88

M

Media Task Flags 92
MediaChangedNonPrimarySource function 14
MediaCompare function 14
MediaCurrentMediaQueuedData function 15
MediaDisposeTargetRefCon function 16
MediaDoIdleActions function 16
MediaEmptyAllPurgeableChunks function 17
MediaEmptySampleCache function 17
MediaEnterEmptyEdit function 18
MediaEQSpectrumBandsRecord structure 88
MediaEQSpectrumBandsRecordPtr data type 88
MediaFlushNonPrimarySourceData function 18
MediaForceUpdate function 19
MediaForceUpdate Values 90
MediaGetActionsForQTEvent function 19
MediaGetChunkManagementFlags function 20
MediaGetClock function 21
MediaGetDrawingRgn function 21
MediaGetEffectiveSoundBalance function 22
MediaGetEffectiveVolume function 22
MediaGetErrorString function 23
MediaGetGraphicsMode function 24
MediaGetInvalidRegion function 24
MediaGetMediaInfo function 25
MediaGetMediaLoadState function 25
MediaGetName function 26
MediaGetNextBoundsChange function 27
MediaGetNextStepTime function 28
MediaGetOffscreenBufferSize function 28
MediaGetPublicInfo function 29
MediaGetPurgeableChunkMemoryAllowance function 30
MediaGetSampleDataPointer function 31
MediaGetSoundBalance function 31
MediaGetSoundBassAndTreble function 32
MediaGetSoundEqualizerBandLevels function 33
MediaGetSoundEqualizerBands function 33
MediaGetSoundLevelMeterInfo function 34
MediaGetSoundLevelMeteringEnabled function 34
MediaGetSoundOutputComponent function 35
MediaGetSrcRgn function 35
MediaGetTrackOpaque function 36
MediaGetURLLink function 37
MediaGetUserPreferredCodecs function 38
MediaGetVideoParam function 38
MediaGGetIdleManager function 39
MediaGGetStatus function 40
MediaGSetActiveSegment function 40
MediaGSetIdleManager function 41
MediaGSetVolume function 42
MediaHasCharacteristic function 42
MediaHitTestForTargetRefCon function 43
MediaHitTestTargetRefCon function 44
MediaHitTestTargetRefCon Values 93
MediaIdle function 45
MediaInitialize function 47
MediaInvalidateRegion function 48

MediaMakeMediaTimeTable function 48
MediaMCIsPlayerEvent function 50
MediaNavigateTargetRefCon function 50
MediaNavigateTargetRefCon Values 92
MediaPrePrerollBegin function 51
MediaPrePrerollCancel function 52
MediaPreroll function 52
MediaPutMediaInfo function 53
MediaQueueNonPrimarySourceData function 54
MediaRefConGetProperty function 55
MediaRefConSetProperty function 56
MediaRefConSetProperty Values 92
MediaReleaseSampleDataPointer function 56
MediaResolveTargetRefCon function 57
MediaSampleDescriptionB2N function 58
MediaSampleDescriptionChanged function 58
MediaSampleDescriptionN2B function 59
MediaSetActionsCallback function 59
MediaSetActive function 60
MediaSetChunkManagementFlags function 61
MediaSetChunkManagementFlags Values 91
MediaSetClip function 61
MediaSetDimensions function 62
MediaSetDoMCActionCallback function 63
MediaSetGraphicsMode function 63
MediaSetGWorld function 64
MediaSetHandlerCapabilities function 65
MediaSetHints function 66
MediaSetMatrix function 67
MediaSetMediaTimeScale function 67
MediaSetMovieTimeScale function 68
MediaSetNonPrimarySourceData function 69
MediaSetPublicInfo function 71
MediaSetPurgeableChunkMemoryAllowance function 72
MediaSetRate function 72
MediaSetScreenLock function 73
MediaSetSoundBalance function 74
MediaSetSoundBassAndTreble function 74
MediaSetSoundEqualizerBands function 75
MediaSetSoundLevelMeteringEnabled function 76
MediaSetSoundLocalizationData function 76
MediaSetSoundOutputComponent function 77
MediaSetTrackInputMapReference function 77
MediaSetUserPreferredCodecs function 78
MediaSetVideoParam function 79
MediaSetVideoParam Values 91
MediaTargetRefConsEqual function 80
MediaTimeBaseChanged function 81
MediaTrackEdited function 81
MediaTrackPropertyAtomChanged function 82
MediaTrackReferencesChanged function 82
MediaVideoOutputChanged function 83

N

NewPrePrerollCompleteUPP function 83

P

PrePrerollCompleteProc callback 84
PrePrerollCompleteUPP data type 89

Q

QTCustomActionTargetPtr data type 89
QTCustomActionTargetRecord structure 89