Component Creation Reference for QuickTime

QuickTime > QuickTime Component Creation

2006-05-23

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

Simultaneously published in the United States and Canada.

Contents

Component Creation Reference for QuickTime

Framework:	Frameworks/QuickTime.framework
Declared in	Movies.h QuickTimeComponents.h

Overview

APIs are provided to help developer create new components that import and export data to and from QuickTime movies, including managing movie previews.

Functions by Task

Compressing Image Sequences

`SCCompressSequenceBegin` (page 81)
Initiates a sequence-compression operation.

`SCCompressSequenceEnd` (page 82)
Ends a sequence-compression operation.

`SCCompressSequenceFrame` (page 82)
Continues a sequence-compression operation.

Compressing Still Images

`SCCompressImage` (page 78)
Compresses an image that is stored in a PixMap structure.

`SCCompressPicture` (page 79)
Compresses a Picture structure that is stored by a handle.

`SCCompressPictureFile` (page 80)
Compresses a Picture structure that is stored in a file.

Configuring Movie Data Export Components

`MovieExportDoUserDialog` (page 32)
Requests that a component display its user dialog box.

`MovieExportSetProgressProc` (page 40)
Assigns a movie progress function.

Configuring Movie Data Import Components

`MovieImportDoUserDialog` (page 48)
Requests that a component display its user dialog box.

`MovieImportSetAuxiliaryData` (page 59)
Provides additional data to a component.

`MovieImportSetChunkSize` (page 60)
The amount of data a component works with at a time.

`MovieImportSetDimensions` (page 61)
Specifies a new track's spatial dimensions.

`MovieImportSetDuration` (page 62)
Controls the duration of the data that a component pastes into the target movie.

`MovieImportSetFromScrap` (page 63)
Indicates that the source data resides on the scrap.

`MovieImportSetMediaFile` (page 65)
Specifies a media file that is to receive the imported movie data.

`MovieImportSetProgressProc` (page 67)
Assigns a movie progress function.

`MovieImportSetSampleDescription` (page 68)
Provides a SampleDescription structure to a movie data import component.

`MovieImportSetSampleDuration` (page 69)
Sets the sample duration for new samples to be created with a component.

Creating a Compression Graphics World

`SCNewGWorld` (page 92)
Creates a graphics world based on the current compression settings.

Creating Previews

`PreviewMakePreview` (page 75)
Creates previews by allocating a handle to data that is to be added to a file.

`PreviewMakePreviewReference` (page 76)
Returns the type and identification number of a resource within a file to be used as the preview for a file.

Displaying Previews

`PreviewShowData` (page 77)
Displays a preview if it does not handle events.

Displaying the Standard Image-Compression Dialog Box

`SCRequestImageSettings` (page 94)
Displays the standard image dialog box to the user and shows default settings you have established.

`SCRequestSequenceSettings` (page 95)
Displays the standard sequence dialog box to the user and shows default settings you have established.

Exporting Movie Data

`MovieExportAddDataSource` (page 30)
Defines a data source for use with an export operation performed by MovieExportFromProceduresToDataRef.

`MovieExportDisposeGetDataAndPropertiesProcs` (page 31)
Disposes of the memory associated with the procedures returned by MovieExportNewGetDataAndPropertiesProcs.

`MovieExportFromProceduresToDataRef` (page 33)
Exports data provided by MovieExportAddDataSource to a specified location.

`MovieExportGetAuxiliaryData` (page 34)
Retrieves additional data from a component.

`MovieExportGetSettingsAsAtomContainer` (page 35)
Retrieves the current settings from the movie export component.

`MovieExportNewGetDataAndPropertiesProcs` (page 37)
Returns MovieExportGetPropertyProc and MovieExportGetDataProc callbacks that can be passed to MovieExportAddDataSource to create a new data source.

`MovieExportSetGetMoviePropertyProc` (page 39)
Specifies the procedure that the export component should call to retrieve movie level properties during MovieExportFromProceduresToDataRef.

`MovieExportSetSampleDescription` (page 40)
Requests the format of the exported data.

`MovieExportSetSettingsFromAtomContainer` (page 41)
Sets the movie export component's current configuration from passed settings data.

`MovieExportToDataRef` (page 43)
Allows an application to request that data be exported to a data reference instead of to a file.

`MovieExportToFile` (page 44)
Exports data to a file, using a movie data export component.

`MovieExportToHandle` (page 45)
Exports data from a movie, using a movie data export component.

`MovieExportValidate` (page 46)
Determines whether a movie export component can export all the data for a specified movie or track.

Exporting Text

`TextExportGetDisplayData` (page 110)
Retrieves text display information for the current sample in the specified text export component.

`TextExportGetSettings` (page 110)
Retrieves the value of the text export option for the specified text export component.

`TextExportGetTimeFraction` (page 111)
Retrieves the time scale the specified text export component uses to calculate time stamps.

`TextExportSetSettings` (page 112)
Sets the value of the text export option for the specified text export component.

`TextExportSetTimeFraction` (page 112)
Sets the time scale the specified text export component uses to calculate time stamps.

Getting Default Settings for an Image or a Sequence

`SCDefaultPictFileSettings` (page 86)
Derives default compression settings for a Picture structure that is stored in a file.

`SCDefaultPictHandleSettings` (page 86)
Derives default compression settings for a Picture structure that is stored by a handle.

`SCDefaultPixMapSettings` (page 87)
Derives default compression settings for an image that is stored in a pixel map.

Handling Preview Events

`PreviewEvent` (page 75)
May be called as appropriate if a preview component handles events.

Importing MIDI Files

`MIDIImportGetSettings` (page 28)
Obtains settings that control the importation of MIDI files.

`MIDIImportSetSettings` (page 29)
Define settings that control the importation of MIDI files.

Importing Movie Data

`MovieImportFile` (page 50)
Imports data from a file, using a movie data import component.

`MovieImportGetAuxiliaryDataType` (page 52)
Returns the type of the auxiliary data that a component can accept.

`MovieImportGetDestinationMediaType` (page 53)
Returns the current type of a movie importer's destination media.

`MovieImportGetFileType` (page 54)
Allows your movie data import component to tell the Movie Toolbox the appropriate file type for the most-recently imported movie file.

`MovieImportGetMIMETypeList` (page 56)
Returns a list of MIME types supported by the movie import component.

`MovieImportGetSettingsAsAtomContainer` (page 57)
Retrieves the current settings from the movie import component.

`MovieImportHandle` (page 57)
Imports data from a handle, using a movie data import component.

`MovieImportSetOffsetAndLimit` (page 66)
Specifies location and size of data that should be imported.

`MovieImportSetOffsetAndLimit64` (page 67)
Specifies location and size of data that should be imported from a file.

`MovieImportSetSettingsFromAtomContainer` (page 69)
Sets the movie import component's current configuration from the passed settings data.

`MovieImportValidate` (page 70)
Allows your movie data import component to validate the data to be passed to your component.

`MovieImportValidateDataRef` (page 71)
Validates the data file indicated by the data reference.

Managing the Time

`ClockRateChanged` (page 22)
In a clock component, is called whenever the callback's time base rate changes.

`ClockSetTimeBase` (page 23)
In a clock component, is called when an application creates a time base that uses the clock component.

`ClockStartStopChanged` (page 23)
In a clock component, is called whenever the start or stop time of the callback's time base changes.

`ClockTimeChanged` (page 24)
In a clock component, is called whenever the callback's time base time value is set.

Movie Functions

`MovieImportSetNewMovieFlags` (page 65)
Implemented by a movie import component to determine the original flags for NewMovieFromDataRef.

Positioning Dialog Boxes and Rectangles

`SCGetBestDeviceRect` (page 88)
Determines the boundary rectangle that surrounds the display device that supports the largest color or grayscale palette.

`SCPositionDialog` (page 93)
Helps position a dialog box on the screen.

`SCPositionRect` (page 93)
Positions a rectangle on the screen.

Specifying a Test Image

`SCSetTestImagePictFile` (page 98)
Sets the dialog box's test image from a Picture structure that is stored in a picture file.

`SCSetTestImagePictHandle` (page 99)
Sets the dialog box's test image from a Picture structure that is stored in a handle.

`SCSetTestImagePixMap` (page 100)
Sets the dialog box's test image from a Picture structure that is stored in a PixMap structure.

Tween Component Requirements

`TweenerDoTween` (page 113)
Performs a tween operation.

`TweenerInitialize` (page 114)
Initializes your tween component for a single tween operation.

`TweenerReset` (page 115)
Cleans up when the tween operation is finished.

Using Callback Functions

`ClockCallMeWhen` (page 17)
In a clock component, schedules a callback event for invocation.

`ClockCancelCallBack` (page 18)
In a clock component, removes the specified callback event from the list of scheduled callback events for a time base.

`ClockDisposeCallBack` (page 19)
In a clock component, disposes of the memory associated with the specified callback event.

`ClockNewCallBack` (page 21)
In a clock component, allocates memory for a new callback event.

Working With Image or Sequence Settings

`SCGetInfo` (page 90)
Retrieves configuration information from the standard dialog component.

`SCSetInfo` (page 96)
Modifies the standard dialog component's configuration information.

Working With The Idle Manager

`MovieImportSetIdleManager` (page 63)
Lets a movie importer report its idling needs.

Working With the Timecode Media Handler

`TCFrameNumberToTimeCode` (page 102)
Converts a frame number into its corresponding timecode time value.

`TCGetCurrentTimeCode` (page 102)
Retrieves the timecode and source identification information for the current movie time.

`TCGetDisplayOptions` (page 103)
Retrieves the text characteristics that apply to timecode information displayed in a movie.

`TCGetSourceRef` (page 104)
Retrieves the source information from the timecode media sample reference.

`TCGetTimeCodeAtTime` (page 104)
Returns a track's timecode information corresponding to a specific media time.

`TCGetTimeCodeFlags` (page 105)
Retrieves the timecode control flags.

`TCSetDisplayOptions` (page 106)
Sets the text characteristics that apply to timecode information displayed in a movie.

`TCSetSourceRef` (page 107)
Changes the source information in the timecode media sample reference.

`TCSetTimeCodeFlags` (page 107)
Changes the flag that affects how the toolbox handles timecode information.

`TCTimeCodeToFrameNumber` (page 108)
Converts a timecode time value into its corresponding frame number.

`TCTimeCodeToString` (page 109)
Converts a time value into a text string (HH:MM:SS:FF).

Supporting Functions

`ClockGetRate` (page 20)
Fetches the rate of a specified clock.

`ClockGetRateChangeConstraints` (page 20)
Obtains minimum and maximum delays that a clock could introduce during a rate change.

`ClockGetTime` (page 21)
Obtains the current time according to a specified clock.

`DisposeMovieExportGetDataUPP` (page 25)
Disposes of a MovieExportGetDataUPP pointer.

`DisposeMovieExportGetPropertyUPP` (page 25)
Disposes of a MovieExportGetPropertyUPP pointer.

`DisposeMovieExportStageReachedCallbackUPP` (page 26)
Disposes of a MovieExportStageReachedCallbackUPP pointer.

`DisposeSCModalFilterUPP` (page 26)
Disposes of an SCModalFilterUPP pointer.

`DisposeSCModalHookUPP` (page 27)
Disposes of an SCModalHookUPP pointer.

GraphicsImageImportGetSequenceEnabled (page 27)
Undocumented

GraphicsImageImportSetSequenceEnabled (page 28)
Undocumented

MovieExportGetCreatorType (page 34)
Undocumented

MovieExportGetFileNameExtension (page 35)
Undocumented

MovieExportGetShortFileTypeString (page 36)
Undocumented

MovieExportGetSourceMediaType (page 37)
Returns either the track type if a movie export component is track-specific or 0 if it is track-independent.

MovieImportDataRef (page 47)
Undocumented

MovieImportDoUserDialogDataRef (page 49)
Requests that a movie import component display its user dialog box.

MovieImportEstimateCompletionTime (page 50)
Undocumented

MovieImportGetDontBlock (page 53)
Undocumented

MovieImportGetLoadState (page 54)
Undocumented

MovieImportGetMaxLoadedTime (page 55)
Undocumented

MovieImportGetSampleDescription (page 56)
Gets the current sample description for a movie import component.

MovieImportIdle (page 59)
Undocumented

MovieImportSetDontBlock (page 61)
Undocumented

MovieImportSetMediaDataRef (page 64)
Specifies a storage location that is to receive imported movie data.

NewMovieExportGetDataUPP (page 72)
Allocates a Universal Procedure Pointer for the MovieExportGetDataProc callback.

NewMovieExportGetPropertyUPP (page 73)
Allocates a Universal Procedure Pointer for the MovieExportGetPropertyProc callback.

NewMovieExportStageReachedCallbackUPP (page 73)
Allocates a new Universal Procedure Pointer for a MovieExportStageReachedCallbackProc callback.

NewSCModalFilterUPP (page 74)
Allocates a Universal Procedure Pointer for the SCModalFilterProc callback.

NewSCModalHookUPP (page 74)
Allocates a Universal Procedure Pointer for the SCModalHookProc callback.

`SCAsyncIdle` (page 78)
: Called occasionally while performing asynchronous compression with SCCompressSequenceFrameAsync.

`SCAudioInvokeLegacyCodecOptionsDialog` (page 78)
: Invokes the legacy code options dialog of an audio codec component.

`SCCompressSequenceFrameAsync` (page 84)
: An asynchronous variant of SCCompressSequenceFrame, with a completion callback.

`SCCopyCompressionSessionOptions` (page 85)
: Creates a compression session options object based upon the settings in the Standard Compression component.

`SCGetCompressFlags` (page 88)
: Gets compression flags for a standard image-compression dialog component.

`SCGetCompressionExtended` (page 89)
: Undocumented

`SCGetSettingsAsAtomContainer` (page 91)
: Places the current configuration from the standard image-compression component in a QT atom container.

`SCGetSettingsAsText` (page 91)
: Undocumented

`SCSetCompressFlags` (page 96)
: Sets compression flags for a standard image-compression dialog component.

`SCSetSettingsFromAtomContainer` (page 97)
: Sets the standard image-compression component's current configuration from data in a QT atom container.

Functions

ClockCallMeWhen

In a clock component, schedules a callback event for invocation.

```
ComponentResult ClockCallMeWhen (
   ComponentInstance aClock,
   QTCallBack cb,
   long param1,
   long param2,
   long param3
);
```

Parameters

aClock
: Specifies the clock for the operation. Applications obtain this identifier from `OpenComponent`.

cb
: Specifies the callback event for the operation. The Movie Toolbox obtains this value from your component's `ClockNewCallBack` (page 21) function.

param1

Contains data supplied to the Movie Toolbox in the param1 parameter to the `CallMeWhen` function. Your component interprets this parameter based on the value of the `callBackType` parameter to the `ClockNewCallBack` (page 21) function. If `callBackType` is set to `callBackAtTime`, the param1 parameter contains flags (see below) indicating when to invoke your callback function for this callback event. If the `callBackType` parameter is set to `callBackAtRate`, param1 contains flags (see below) indicating when to invoke your callback function for this event.

param2

Contains data supplied to the Movie Toolbox in the param2 parameter to the `CallMeWhen` function. Your component interprets this parameter based on the value of the `callBackType` parameter to the `ClockNewCallBack` (page 21) function. If `callBackType` is set to `callBackAtTime`, the param2 parameter contains the time value at which your callback function is to be invoked for this event. The param1 parameter contains flags affecting when the Movie Toolbox calls your function. If `callBackType` is set to `callBackAtRate`, the param2 parameter contains the rate value at which your callback function is to be invoked for this event.

param3

Contains data supplied to the Movie Toolbox in the param3 parameter to the `CallMeWhen` function. If `cbType` is set to `callBackAtTime`, param3 contains the time scale in which to interpret the time value that is stored in param2.

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

Discussion
If your clock component successfully schedules the callback event, you should call the `AddCallBackToTimeBase` function to add it to the list of callback events for the corresponding time base. If your component cannot schedule the callback event, it should return an appropriate error.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`QuickTimeComponents.h`

ClockCancelCallBack

In a clock component, removes the specified callback event from the list of scheduled callback events for a time base.

```
ComponentResult ClockCancelCallBack (
   ComponentInstance aClock,
   QTCallBack cb
);
```

Parameters

aClock

Specifies the clock for the operation. Your application obtains this identifier from the Component Manager's `OpenComponent` function.

cb

Specifies the callback event for the operation. The Movie Toolbox obtains this value from your component's `ClockNewCallBack` (page 21) function.

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

Discussion
If your clock component successfully cancels the callback event, you should call the `RemoveCallBackFromTimeBase` function so that the Movie Toolbox can remove the callback event from its list of scheduled events.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`QuickTimeComponents.h`

ClockDisposeCallBack

In a clock component, disposes of the memory associated with the specified callback event.

```
ComponentResult ClockDisposeCallBack (
   ComponentInstance aClock,
   QTCallBack cb
);
```

Parameters

aClock

Specifies the clock for the operation. Applications obtain this identifier from the Component Manager's `OpenComponent` function.

cb

Specifies the callback event for the operation. The Movie Toolbox obtains this value from your component's `ClockNewCallBack` (page 21) function.

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

Discussion
You should not call this function at interrupt time.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`QuickTimeComponents.h`

ClockGetRate

Fetches the rate of a specified clock.

```
ComponentResult ClockGetRate (
   ComponentInstance aClock,
   Fixed *rate
);
```

Parameters

aClock

Specifies the clock for the operation. Applications obtain this identifier from the Component Manager's `OpenComponent` function.

rate

Pointer to memory where the clock rate is returned.

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
`QuickTimeComponents.h`

ClockGetRateChangeConstraints

Obtains minimum and maximum delays that a clock could introduce during a rate change.

```
ComponentResult ClockGetRateChangeConstraints (
   ComponentInstance aClock,
   TimeRecord *minimumDelay,
   TimeRecord *maximumDelay
);
```

Parameters

aClock

Specifies the clock for the operation. Applications obtain this identifier from `OpenComponent`.

minimum

A pointer to a `TimeRecord` structure that the clock will update with the minimum delay introduced during a rate change. You can pass `NIL` if you do not want to receive this information.

maximum

A pointer to a `TimeRecord` structure that the clock will update with the maximum delay introduced during a rate change. You can pass `NIL` if you do not want to receive this information.

Return Value
See `Error Codes` in the QuickTime API Reference. Returns `noErr` if there is no error. Returns `badComponentSelector` if the component does not support the call.

Version Notes
Introduced in QuickTime 6.4.

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

Declared In
`QuickTimeComponents.h`

ClockGetTime

Obtains the current time according to a specified clock.

```
ComponentResult ClockGetTime (
   ComponentInstance aClock,
   TimeRecord *out
);
```

Parameters

aClock

Specifies the clock for the operation. You obtain this identifier from `OpenComponent` or `OpenDefaultComponent`.

out

A pointer to a `TimeRecord` structure. The clock component updates this structure with the current time information. Specifically, the clock component sets the `value` field and the `scale` field in the time structure. Your clock component should always return values in its native time scale. This time scale does not change during the life of the component connection.

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
`QuickTimeComponents.h`

ClockNewCallBack

In a clock component, allocates memory for a new callback event.

```
QTCallBack ClockNewCallBack (
   ComponentInstance aClock,
   TimeBase tb,
   short callBackType
);
```

Parameters

aClock

Specifies the clock for the operation. Applications obtain this identifier from the Component Manager's `OpenComponent` function.

tb

Specifies the callback event's time base. Typically, your component does not need to save this specification. You can use the Movie Toolbox's `GetCallBackTimeBase` function to determine the callback event's time base when it is invoked. For more information about time bases, see *Inside Macintosh: QuickTime*.

callBackType

Contains a constant (see below) that specifies when the callback event is to be invoked. The value of this parameter governs how your component interprets the data supplied in the param1, param2, and param3 parameters to `ClockCallMeWhen` (page 17). See these constants:

```
callBackAtTime
callBackAtRate
callBackAtTimeJump
callBackAtInterrupt
```

Return Value
A pointer to a `CallBackRecord` structure. Your software can pass this structure to other functions, such as `ClockRateChanged` (page 22).

Discussion
Your component allocates the memory required to support the callback event. The memory must be in a locked block and must begin with a `QTCallBackHeader` structure initialized to 0. Your component can allocate an arbitrarily large piece of memory for the callback event.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`QuickTimeComponents.h`

ClockRateChanged

In a clock component, is called whenever the callback's time base rate changes.

```
ComponentResult ClockRateChanged (
   ComponentInstance aClock,
   QTCallBack cb
);
```

Parameters

aClock

Specifies the clock for the operation. Applications obtain this identifier from the Component Manager's `OpenComponent` function.

cb

Specifies the callback for the operation. The Movie Toolbox obtains this value from your component's `ClockNewCallBack` (page 21) function.

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

Discussion
The Movie Toolbox calls this function once for each qualified callback function associated with the time base. Note that the Movie Toolbox calls this function only for callback events that are currently scheduled.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`QuickTimeComponents.h`

ClockSetTimeBase

In a clock component, is called when an application creates a time base that uses the clock component.

```
ComponentResult ClockSetTimeBase (
   ComponentInstance aClock,
   TimeBase tb
);
```

Parameters

aClock

Specifies the clock for the operation. Applications obtain this identifier from the Component Manager's `OpenComponent` function.

tb

Specifies the time base that is associated with the clock.

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
`QuickTimeComponents.h`

ClockStartStopChanged

In a clock component, is called whenever the start or stop time of the callback's time base changes.

```
ComponentResult ClockStartStopChanged (
   ComponentInstance aClock,
   QTCallBack cb,
   Boolean startChanged,
   Boolean stopChanged
);
```

Parameters

aClock

Specifies the clock for the operation. Applications obtain this identifier from the Component Manager's `OpenComponent` function.

cb

Specifies the callback for the operation. The Movie Toolbox obtains this value from your component's `ClockNewCallBack` (page 21) function.

startChanged

Indicates that the start time of the time base associated with the clock component instance has changed.

stopChanged

Indicates that the stop time of the time base associated with the clock component instance has changed.

Return Value

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

Discussion

The Movie Toolbox calls this function once for each qualified callback function associated with the time base. Note that the Movie Toolbox calls this function only for callback events that are currently scheduled.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`QuickTimeComponents.h`

ClockTimeChanged

In a clock component, is called whenever the callback's time base time value is set.

```
ComponentResult ClockTimeChanged (
   ComponentInstance aClock,
   QTCallBack cb
);
```

Parameters

aClock

Specifies the clock for the operation. Applications obtain this identifier from the Component Manager's `OpenComponent` function.

cb

Specifies the callback for the operation. The Movie Toolbox obtains this value from your component's `ClockNewCallBack` (page 21) function.

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
`QuickTimeComponents.h`

DisposeMovieExportGetDataUPP

Disposes of a MovieExportGetDataUPP pointer.

```
void DisposeMovieExportGetDataUPP (
   MovieExportGetDataUPP userUPP
);
```

Parameters

`userUPP`

A `MovieExportGetDataUPP` 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 v10.0 and later.

Related Sample Code
CIVideoDemoGL
qtmoviefromprocs
qtmoviefromprocs.win

Declared In
`QuickTimeComponents.h`

DisposeMovieExportGetPropertyUPP

Disposes of a MovieExportGetPropertyUPP pointer.

```
void DisposeMovieExportGetPropertyUPP (
   MovieExportGetPropertyUPP userUPP
);
```

Parameters

`userUPP`

A `MovieExportGetPropertyUPP` 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 v10.0 and later.

Related Sample Code
CIVideoDemoGL
qtmoviefromprocs
qtmoviefromprocs.win

Declared In
`QuickTimeComponents.h`

DisposeMovieExportStageReachedCallbackUPP

Disposes of a MovieExportStageReachedCallbackUPP pointer.

```
void DisposeMovieExportStageReachedCallbackUPP (
   MovieExportStageReachedCallbackUPP userUPP
);
```

Parameters

userUPP

A `MovieExportStageReachedCallbackUPP` pointer.

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

Declared In
`QuickTimeComponents.h`

DisposeSCModalFilterUPP

Disposes of an SCModalFilterUPP pointer.

```
void DisposeSCModalFilterUPP (
   SCModalFilterUPP userUPP
);
```

Parameters

userUPP

An `SCModalFilterUPP` 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 v10.0 and later.

Related Sample Code
qtcompress
qtcompress.win

Declared In
`QuickTimeComponents.h`

DisposeSCModalHookUPP

Disposes of an SCModalHookUPP pointer.

```
void DisposeSCModalHookUPP (
   SCModalHookUPP userUPP
);
```

Parameters

userUPP

An `SCModalHookUPP` 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 v10.0 and later.

Related Sample Code
qtcompress
qtcompress.win

Declared In
`QuickTimeComponents.h`

GraphicsImageImportGetSequenceEnabled

Undocumented

```
ComponentResult GraphicsImageImportGetSequenceEnabled (
   GraphicImageMovieImportComponent ci,
   Boolean *enable
);
```

Parameters

ci

The component instance that identifies your connection to the movie importer component.

enable

A pointer to a Boolean that returns TRUE if enabled, 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
`QuickTimeComponents.h`

GraphicsImageImportSetSequenceEnabled

Undocumented

```
ComponentResult GraphicsImageImportSetSequenceEnabled (
   GraphicImageMovieImportComponent ci,
   Boolean enable
);
```

Parameters

ci

The component instance that identifies your connection to the movie importer component.

enable

Pass TRUE to enable, FALSE to disable.

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
Graphic Import-Export
ImproveYourImage

Declared In
`QuickTimeComponents.h`

MIDIImportGetSettings

Obtains settings that control the importation of MIDI files.

```
ComponentResult MIDIImportGetSettings (
   TextExportComponent ci,
   long *setting
);
```

Parameters

ci

A text export component instance used to import a MIDI file. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

setting

Flags (see below) that control the importation of MIDI files. The flags correspond to the checkboxes in the MIDI Import Options dialog box. See these constants:

`kMIDIImportSilenceBefore`

`kMIDIImportSilenceAfter`

`kMIDIImport2OPlayable`

`kMIDIImportWantLyrics`

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
`QuickTimeComponents.h`

MIDIImportSetSettings

Define settings that control the importation of MIDI files.

```
ComponentResult MIDIImportSetSettings (
   TextExportComponent ci,
   long setting
);
```

Parameters

ci

A text export component instance used to import a MIDI file. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

setting

Flags (see below) that control the importation of MIDI files. The flags correspond to the checkboxes in the MIDI Import Options dialog box. See these constants:

`kMIDIImportSilenceBefore`

`kMIDIImportSilenceAfter`

`kMIDIImport2OPlayable`

`kMIDIImportWantLyrics`

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
`QuickTimeComponents.h`

MovieExportAddDataSource

Defines a data source for use with an export operation performed by MovieExportFromProceduresToDataRef.

```
ComponentResult MovieExportAddDataSource (
   MovieExportComponent ci,
   OSType trackType,
   TimeScale scale,
   long *trackID,
   MovieExportGetPropertyUPP getPropertyProc,
   MovieExportGetDataUPP getDataProc,
   void *refCon
);
```

Parameters

ci

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

trackType

The type of media provided by this data source. This normally corresponds to a QuickTime media type such as `VideoMediaType` or `SoundMediaType`.

scale

The time scale for time values passed to `getDataProc` parameter. If the source data is being taken from a QuickTime track, this value is typically the media's time scale.

trackID

An identifier for the data source. This identifier is returned from the call.

getPropertyProc

A `MovieExportGetPropertyProc` callback that provides information about processing source samples.

getDataProc

A `MovieExportGetDataProc` callback the export component uses to request sample data.

refCon

Passed to the procedures specified in the `getPropertyProc` and `getDataProc` parameters. Use this parameter to point to a data structure containing any information your callbacks need.

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

Discussion
Before starting an export operation, all the data sources must be defined by calling this function once for each data source.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
CIVideoDemoGL
ElectricImageComponent
ElectricImageComponent.win
qtmoviefromprocs
qtmoviefromprocs.win

Declared In
`QuickTimeComponents.h`

MovieExportDisposeGetDataAndPropertiesProcs

Disposes of the memory associated with the procedures returned by MovieExportNewGetDataAndPropertiesProcs.

```
ComponentResult MovieExportDisposeGetDataAndPropertiesProcs (
   MovieExportComponent ci,
   MovieExportGetPropertyUPP getPropertyProc,
   MovieExportGetDataUPP getDataProc,
   void *refCon
);
```

Parameters

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

getPropertyProc
: A `MovieExportGetPropertyProc` callback that provides information about processing source samples.

getDataProc
: A `MovieExportGetDataProc` callback that the export component uses to request sample data.

refCon
: Passed to the procedures specified in the `getPropertyProc` and `getDataProc` parameters. Use this parameter to point to a data structure containing any information your callbacks need.

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
CIVideoDemoGL

ElectricImageComponent
ElectricImageComponent.win

Declared In
`QuickTimeComponents.h`

MovieExportDoUserDialog

Requests that a component display its user dialog box.

```
ComponentResult MovieExportDoUserDialog (
   MovieExportComponent ci,
   Movie theMovie,
   Track onlyThisTrack,
   TimeValue startTime,
   TimeValue duration,
   Boolean *canceled
);
```

Parameters

ci

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

theMovie

The movie containing the data to be exported.

onlyThisTrack

Specifies that the export component should only attempt to export the data from a single track. If this parameter is set to `NIL`, the exporter should attempt to export the entire movie, or all of the tracks in the movie that it can export. For example, an audio export component might export multiple audio tracks, mixing them if necessary. If this parameter is not `NIL`, the exporter should attempt to export only the specified track.

startTime

The movie time at which to begin the export operation. If you pass 0, the operation should start at the beginning of the movie or track.

duration

The duration, in movie timescale units, of the segment to be exported. To export the entire movie, or an entire track, pass in the value returned by `GetMovieDuration` or `GetTrackDuration`, minus the value passed in `startTime`, as described above.

canceled

A pointer to a Boolean value. Your component should set this value to TRUE if the user cancels the dialog box, otherwise FALSE. If the user cancels the dialog box, your component should revert to its settings as they were before executing this function.

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
Graphic Import-Export
qtdataexchange
qtmoviefromprocs
qtmoviefromprocs.win
ThreadsExportMovie

Declared In
`QuickTimeComponents.h`

MovieExportFromProceduresToDataRef

Exports data provided by MovieExportAddDataSource to a specified location.

```
ComponentResult MovieExportFromProceduresToDataRef (
   MovieExportComponent ci,
   Handle dataRef,
   OSType dataRefType
);
```

Parameters

ci

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

dataRef

The data reference for the export operation.

dataRefType

The type identifier for the data reference specified by `dataRef`.

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

Discussion
This function exports data provided by `MovieExportAddDataSource` (page 30) to a location specified by `dataRef` and `dataRefType`. Typically `dataRef` contains a Macintosh file alias and `dataRefType` is set to `rAliasType`.

Special Considerations

Movie data export components that support export operations from procedures must set the `canMovieExportFromProcedures` flag in their component flags.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
CIVideoDemoGL
ElectricImageComponent
ElectricImageComponent.win
qtmoviefromprocs

qtmoviefromprocs.win

Declared In
QuickTimeComponents.h

MovieExportGetAuxiliaryData

Retrieves additional data from a component.

```
ComponentResult MovieExportGetAuxiliaryData (
   MovieExportComponent ci,
   Handle dataH,
   OSType *handleType
);
```

Parameters

ci

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

dataH

A handle that is to be filled with the additional data. Your component should resize this handle as appropriate. Your component is not responsible for disposing of this handle.

handleType

A pointer to the type of data you place in the handle specified by the `data` parameter.

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

Discussion
Your component should expect the application to call this function after the export process ends.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
QuickTimeComponents.h

MovieExportGetCreatorType

Undocumented

```
ComponentResult MovieExportGetCreatorType (
   MovieExportComponent ci,
   OSType *creator
);
```

Parameters

ci

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

creator

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
`QuickTimeComponents.h`

MovieExportGetFileNameExtension

Undocumented

```
ComponentResult MovieExportGetFileNameExtension (
   MovieExportComponent ci,
   OSType *extension
);
```

Parameters

ci

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

extension

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
`QuickTimeComponents.h`

MovieExportGetSettingsAsAtomContainer

Retrieves the current settings from the movie export component.

```
ComponentResult MovieExportGetSettingsAsAtomContainer (
   MovieExportComponent ci,
   QTAtomContainer *settings
);
```

Parameters

ci

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

settings

The address where the newly-created atom container should be stored by the call. The caller is responsible for disposing of the returned QT atom container.

Return Value

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

Discussion

Applications can call this function to obtain a correctly formatted atom container to use with `MovieExportSetSettingsFromAtomContainer` (page 41). This might be done after a call to `MovieExportDoUserDialog` (page 32), for example, to apply the user-obtained settings to a series of exports.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

BackgroundExporter

qtdataexchange

qtdataexchange.win

qtmoviefromprocs

qtmoviefromprocs.win

Declared In

`QuickTimeComponents.h`

MovieExportGetShortFileTypeString

Undocumented

```
ComponentResult MovieExportGetShortFileTypeString (
   MovieExportComponent ci,
   Str255 typeString
);
```

Parameters

ci

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

typeString

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
`QuickTimeComponents.h`

MovieExportGetSourceMediaType

Returns either the track type if a movie export component is track-specific or 0 if it is track-independent.

```
ComponentResult MovieExportGetSourceMediaType (
   MovieExportComponent ci,
   OSType *mediaType
);
```

Parameters

ci

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

mediaType

The track type if the component is track-specific or 0 if it is track-independent.

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

Discussion
This routine returns the same values that were previously stored in the `componentManufacturer` field of the `ComponentDescription` structure. This frees up the field to be used for the manufacturer.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`QuickTimeComponents.h`

MovieExportNewGetDataAndPropertiesProcs

Returns MovieExportGetPropertyProc and MovieExportGetDataProc callbacks that can be passed to MovieExportAddDataSource to create a new data source.

```
ComponentResult MovieExportNewGetDataAndPropertiesProcs (
   MovieExportComponent ci,
   OSType trackType,
   TimeScale *scale,
   Movie theMovie,
   Track theTrack,
   TimeValue startTime,
   TimeValue duration,
   MovieExportGetPropertyUPP *getPropertyProc,
   MovieExportGetDataUPP *getDataProc,
   void **refCon
);
```

Parameters

ci

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

trackType

The format of the data to be generated by the returned `MovieExportGetDataProc`.

scale

The time scale returned from this function; this should be passed on to `MovieExportAddDataSource` (page 30) with the procedures.

theMovie

The movie for this operation, supplied by the Movie Toolbox. Your component may use this identifier to obtain sample data from the movie or to obtain information about the movie.

theTrack

The track for this operation. This track identifier is supplied by the Movie Toolbox.

startTime

The starting point of the track or movie segment to be converted. This time value is expressed in the movie's time coordinate system.

duration

The duration of the track or movie segment to be converted. This duration value is expressed in the movie's time coordinate system.

getPropertyProc

A `MovieExportGetPropertyProc` callback that provides information about processing source samples.

getDataProc

A `MovieExportGetDataProc` callback that the export component uses to request sample data.

refCon

Passed to the procedures specified in the `getPropertyProc` and `getDataProc` parameters. Use this parameter to point to a data structure containing any information your callbacks need.

Return Value

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

Discussion

This function exists in order to provide a standard way of getting data using this protocol out of a movie or track. The returned procedures must be disposed by calling `MovieExportDisposeGetDataAndPropertiesProcs` (page 31).

Special Considerations

This function is only implemented by movie data export components.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
CIVideoDemoGL
ElectricImageComponent
ElectricImageComponent.win

Declared In
`QuickTimeComponents.h`

MovieExportSetGetMoviePropertyProc

Specifies the procedure that the export component should call to retrieve movie level properties during MovieExportFromProceduresToDataRef.

```
ComponentResult MovieExportSetGetMoviePropertyProc (
   MovieExportComponent ci,
   MovieExportGetPropertyUPP getPropertyProc,
   void *refCon
);
```

Parameters

ci

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

getPropertyProc

The `MovieExportGetPropertyProc` callback that the export component will call to retrieve movie-level properties.

refCon

The reference value that will be passed to the callback specified by `getPropertyProc`. Use this parameter 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. With QuickTime 4, applications can specify a `MovieExportGetPropertyProc` that will be called to retrieve movie level properties during the exporter's `MovieExportFromProceduresToDataRef` (page 33) execution. This procedure is identical to a data source property procedure except that it is called for movie properties. For example, with QuickTime 4, the QuickTime movie export component calls the procedure to retrieve the time scale for the exported movie. If the property procedure is not specified or doesn't support this property, than the default movie time scale (600) is used.

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

Declared In
QuickTimeComponents.h

MovieExportSetProgressProc

Assigns a movie progress function.

```
ComponentResult MovieExportSetProgressProc (
   MovieExportComponent ci,
   MovieProgressUPP proc,
   long refcon
);
```

Parameters

ci

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

proc

A pointer to the application's `MovieProgressProc` callback. If this parameter is set to `NIL`, the application is removing its progress function. In this case, your component should stop calling the progress function.

refcon

A reference constant. Your component should pass this constant back to the application's progress function whenever you call that function. Use this parameter to point to a data structure containing any information the callback needs.

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

Discussion
These progress functions must support the same interface as Movie Toolbox progress functions. Note that this interface not only allows you to report progress to the application, but also allows the application to cancel the request.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
CIVideoDemoGL
ThreadsExportMovie

Declared In
QuickTimeComponents.h

MovieExportSetSampleDescription

Requests the format of the exported data.

```
ComponentResult MovieExportSetSampleDescription (
   MovieExportComponent ci,
   SampleDescriptionHandle desc,
   OSType mediaType
);
```

Parameters

ci

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

desc

A handle to a valid `SampleDescription` structure.

mediaType

The type of media the `SampleDescription` structure is for. For example, if the sample description was a sound description, this parameter would be set to `SoundMediaType`.

Return Value

See `Error Codes`. Returns `badComponentSelector` if you should be passing a QT atom container (see discussion, below). Returns `noErr` if there is no error.

Discussion

A movie export component may use all, some, or none of the settings from the `SampleDescription` structure.

If your application attempts to set the sample description using this function, and receives the `badComponentSelector` error, you may need to pass in the sample description using `MovieExportSetSettingsFromAtomContainer` (page 41). You can use `MovieExportGetSettingsAsAtomContainer` (page 35) to obtain a correctly formatted atom container to modify.

Special Considerations

This function is not implemented by all movie export components, but is supported by the sound movie export component, for example.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

MovieToAIFF

soundsnippets

soundsnippets.win

Declared In

`QuickTimeComponents.h`

MovieExportSetSettingsFromAtomContainer

Sets the movie export component's current configuration from passed settings data.

```
ComponentResult MovieExportSetSettingsFromAtomContainer (
   MovieExportComponent ci,
   QTAtomContainer settings
);
```

Parameters

ci

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

settings

A QT atom container that contains the settings.

Return Value

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

Discussion

The atom container may contain atoms other than those expected by the particular component type or may be missing certain atoms. This function uses only those settings it understands.

Here is sample code that overrides compression settings:

```
// MovieExportSetSettingsFromAtomContainer coding example
ComponentInstance sc;
QTAtomContainer compressorData;
SCSpatialSettings ss;
sc =OpenDefaultComponent(StandardCompressionType,
                            StandardCompressionSubType);
ss.codecType =kCinepakCodecType;
ss.codec =NIL;
ss.depth =0;
ss.spatialQuality =codecHighQuality
err =SCSetInfo(sc, scSpatialSettingsType, &ss);
err =SCGetSettingsAsAtomContainer(sc, &compressorData);
MovieExportSetSettingsFromAtomContainer (qtvrExport, compressorData);
```

Special Considerations

Some movie export components treat sample descriptions as part of their settings. If your application attempts to set the sample description using `MovieExportSetSampleDescription` (page 40), and receives the `badComponentSelector` error, you may need to pass in the `SampleDescription` structure using this function.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

BackgroundExporter

qtdataexchange

qtdataexchange.win

ThreadsExportMovie

vrmakepano

Declared In
QuickTimeComponents.h

MovieExportToDataRef

Allows an application to request that data be exported to a data reference instead of to a file.

```
ComponentResult MovieExportToDataRef (
   MovieExportComponent ci,
   Handle dataRef,
   OSType dataRefType,
   Movie theMovie,
   Track onlyThisTrack,
   TimeValue startTime,
   TimeValue duration
);
```

Parameters

ci

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

dataRef

A handle to a data reference indicating where the data should be stored.

dataRefType

The type of the data reference. For exporting to a file, the `dataRef` is a Macintosh file alias and the `dataRefType` is `rAliasType`.

theMovie

The movie for this operation. This movie identifier is supplied by the Movie Toolbox. Your component may use this identifier to obtain sample data from the movie or to obtain information about the movie.

onlyThisTrack

Identifies a track that is to be converted. This track identifier is supplied by the Movie Toolbox. If this parameter contains a track identifier, your component must convert only the specified track.

startTime

The starting point of the track or movie segment to be converted. This time value is expressed in the movie's time coordinate system.

duration

The duration of the track or movie segment to be converted. This duration value is expressed in the movie's time coordinate system.

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
CIVideoDemoGL
ElectricImageComponent

ElectricImageComponent.win
ThreadsExportMovie

Declared In
`QuickTimeComponents.h`

MovieExportToFile

Exports data to a file, using a movie data export component.

```
ComponentResult MovieExportToFile (
   MovieExportComponent ci,
   const FSSpec *theFile,
   Movie theMovie,
   Track onlyThisTrack,
   TimeValue startTime,
   TimeValue duration
);
```

Parameters

ci

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

theFile

A pointer to the file that is to receive the converted movie data. This file's type value corresponds to your component's subtype value.

theMovie

The movie for this operation. This movie identifier is supplied by the Movie Toolbox. Your component may use this identifier to obtain sample data from the movie or to obtain information about the movie.

onlyThisTrack

Identifies a track that is to be converted. This track identifier is supplied by the Movie Toolbox. If this parameter contains a track identifier, your component must convert only the specified track.

startTime

The starting point of the track or movie segment to be converted. This time value is expressed in the movie's time coordinate system.

duration

The duration of the track or movie segment to be converted. This duration value is expressed in the movie's time coordinate system.

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

Discussion
The requesting program or Movie Toolbox must create the destination file before calling this function. Furthermore, your component may not destroy any data in the destination file. If you cannot add data to the specified file, return an appropriate error. If your component can write data to a file, be sure to set the `canMovieExportFiles` flag in the `componentFlags` field of your component's `ComponentDescription` structure. Here is an example of using this function with a flattener component:

```
// MovieExportToFile coding example
ComponentDescription desc;
```

```
Component flattener;
ComponentInstance qtvrExport =NIL;
desc.componentType =MovieExportType;
desc.componentSubType =MovieFileType;
desc.componentManufacturer =QTVRFlattenerType;
flattener =FindNextComponent(NIL, &desc);
if (flattener) qtvrExport =OpenComponent (flattener);
if (qtvrExport)
    MovieExportToFile (qtvrExport, &myFileSpec, myQTVRMovie, NIL, 0, 0);
```

Special Considerations

Your component must be prepared to perform this function at any time. You should not expect that any of your component's configuration functions will be called first.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
vrmakepano
VRMakePano Library

Declared In
`QuickTimeComponents.h`

MovieExportToHandle

Exports data from a movie, using a movie data export component.

```
ComponentResult MovieExportToHandle (
   MovieExportComponent ci,
   Handle dataH,
   Movie theMovie,
   Track onlyThisTrack,
   TimeValue startTime,
   TimeValue duration
);
```

Parameters

ci

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

dataH

A handle to be filled with the converted movie data. Your component must write data into this handle that corresponds to your component's subtype value. Your component should resize this handle as appropriate.

theMovie

The movie for this operation. This movie identifier is supplied by the Movie Toolbox. Your component may use this identifier to obtain sample data from the movie or to obtain information about the movie.

onlyThisTrack

Identifies a track that is to be converted. This track identifier is supplied by the Movie Toolbox. If this parameter contains a track identifier, your component must convert only the specified track.

startTime

The starting point of the track or movie segment to be converted. This time value is expressed in the movie's time coordinate system.

duration

The duration of the track or movie segment to be converted. This duration value is expressed in the movie's time coordinate system.

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

Discussion
Your component must be prepared to perform this function at any time. You should not expect that any of your component's configuration functions will be called first. If your component can write data to a handle, be sure to set the `canMovieExportHandles` flag in in the `componentFlags` field of your component's `ComponentDescription` structure.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`QuickTimeComponents.h`

MovieExportValidate

Determines whether a movie export component can export all the data for a specified movie or track.

```
ComponentResult MovieExportValidate (
   MovieExportComponent ci,
   Movie theMovie,
   Track onlyThisTrack,
   Boolean *valid
);
```

Parameters

ci

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

theMovie

The movie to validate.

onlyThisTrack

A track within the movie to validate, or `NIL` if the entire movie is to be validated.

valid

A pointer to a Boolean value. If the data for the movie or track can be exported by the component, the value is TRUE.

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

Discussion
This function allows an application to determine if a particular movie or track could be exported by the specified movie data export component. The movie or track is passed in the `theMovie` and `onlyThisTrack` parameters as they are passed to `MovieExportToFile` (page 44). Although a movie export component can export one or more media types, it may not be able to export all the kinds of data stored in those media. The `MovieExportValidate` function allows applications to get this additional information. Movie data export components that implement this function also set the `canMovieExportValidateMovie` flag in in the `componentFlags` field of their `ComponentDescription` structure.

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
`QuickTimeComponents.h`

MovieImportDataRef

Undocumented

```
ComponentResult MovieImportDataRef (
   MovieImportComponent ci,
   Handle dataRef,
   OSType dataRefType,
   Movie theMovie,
   Track targetTrack,
   Track *usedTrack,
   TimeValue atTime,
   TimeValue *addedDuration,
   long inFlags,
   long *outFlags
);
```

Parameters

ci

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

dataRef

The data reference to the data to be imported.

dataRefType

The type of data reference in the `dataRef` parameter.

theMovie

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

targetTrack

Undocumented

usedTrack
Undocumented

atTime
Undocumented

addedDuration
Undocumented

inFlags
Flags (see below) that control the behavior of this function. See these constants:
movieImportCreateTrack
movieImportInParallel
movieImportMustUseTrack
movieImportWithIdle

outFlags
Flags (see below) that this function sets on return. See these constants:
movieImportResultUsedMultipleTracks
movieImportResultNeedIdles
movieImportResultComplete

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
ElectricImageComponent
ElectricImageComponent.win

Declared In
QuickTimeComponents.h

MovieImportDoUserDialog

Requests that a component display its user dialog box.

```
ComponentResult MovieImportDoUserDialog (
   MovieImportComponent ci,
   const FSSpec *theFile,
   Handle theData,
   Boolean *canceled
);
```

Parameters

ci
A movie data import component instance. Your software obtains this reference from OpenComponent or OpenDefaultComponent.

theFile

A pointer to a valid file specification. If the import request pertains to a file, the application must specify the source file with this parameter and set the `theData` parameter to `NIL`. If the request is for a handle, this parameter is set to `NIL`.

theData

A handle to the data to be imported. If the import request pertains to a handle, the application must specify the source of the `data` with this parameter, and set the `theFile` parameter to `NIL`. If the request is for a file, this parameter is set to `NIL`.

canceled

A pointer to a Boolean value. Your component should set this value to TRUE if the user cancels the dialog box; otherwise, set it to FALSE.

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

Discussion
If your component supports a user dialog box, be sure to set the `hasMovieImportUserInterface` flag in the `componentFlags` field of your component's `ComponentDescription` structure.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
Graphic Import-Export
ImportExportMovie
ImproveYourImage

Declared In
`QuickTimeComponents.h`

MovieImportDoUserDialogDataRef

Requests that a movie import component display its user dialog box.

```
ComponentResult MovieImportDoUserDialogDataRef (
   MovieImportComponent ci,
   Handle dataRef,
   OSType dataRefType,
   Boolean *canceled
);
```

Parameters

ci

The component instance that identifies your connection to the graphics importer component.

dataRef

A data reference that specifies a storage location that contains the data to import.

dataRefType

The type of the data reference.

canceled

A pointer to a Boolean entity that is set to TRUE if the user cancels the export operation.

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

Discussion
This function brings up the option dialog for the import component. The data reference specified the storage location that contains the data to import.

Version Notes
Introduced in QuickTime 6.4.

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

Declared In
`QuickTimeComponents.h`

MovieImportEstimateCompletionTime

Undocumented

```
ComponentResult MovieImportEstimateCompletionTime (
   MovieImportComponent ci,
   TimeRecord *time
);
```

Parameters

ci

A movie data import component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

time

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
`QuickTimeComponents.h`

MovieImportFile

Imports data from a file, using a movie data import component.

```
ComponentResult MovieImportFile (
   MovieImportComponent ci,
   const FSSpec *theFile,
   Movie theMovie,
   Track targetTrack,
   Track *usedTrack,
   TimeValue atTime,
   TimeValue *addedDuration,
   long inFlags,
   long *outFlags
);
```

Parameters

ci

A movie data import component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

theFile

A pointer to the file that contains the data that is to be imported into the `movie`. This file's type value corresponds to your component's subtype value.

theMovie

The movie for this operation. This movie identifier is supplied by the Movie Toolbox. Your component may use this identifier to add sample data to the target movie or to obtain information about the movie.

targetTrack

The track that is to receive the imported data. This track identifier is supplied by the Movie Toolbox and is valid only if the `movieImportMustUseTrack` flag in the `inFlags` parameter is set to 1.

usedTrack

A pointer to the track that received the imported data. Your component returns this track identifier to the Movie Toolbox. Your component needs to set this parameter only if you operate on a single track or if you create a new track. If you modify more than one track, leave the field referred to by this parameter unchanged.

atTime

The time corresponding to the location where your component is to place the imported data. This time value is expressed in the movie's time coordinate system.

addedDuration

A pointer to the duration of the data that your component added to the movie. Your component must specify this value in the movie's time coordinate system.

inFlags

Flags (see below) that specify control information governing the import operation. See these constants:

`movieImportCreateTrack`

`movieImportMustUseTrack`

`movieImportInParallel`

outFlags

Flags (see below) that identify a field that is to receive status information about the import operation. Your component sets the appropriate flags in this field when the operation is complete. See these constants:

`movieImportResultUsedMultipleTracks`

`movieImportInParallel`

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

Discussion
Your component must be prepared to perform this function at any time. You should not expect that any of your component's configuration functions will be called first. If your component can accept data from a file, be sure to set the `canMovieImportFiles` flag in the `componentFlags` field of your component's `ComponentDescription` structure.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
ConvertMovieSndTrack
Graphic Import-Export
ImproveYourImage
SoundPlayer.win
SurfaceVertexProgram

Declared In
`QuickTimeComponents.h`

MovieImportGetAuxiliaryDataType

Returns the type of the auxiliary data that a component can accept.

```
ComponentResult MovieImportGetAuxiliaryDataType (
   MovieImportComponent ci,
   OSType *auxType
);
```

Parameters

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

auxType
A pointer to the type of auxiliary data it can import.

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

Discussion
This function returns the type of the auxiliary data that the `ci` component can accept. For example, calling the text import component with this function indicates that the text import component will use `'styl'` information in addition to `'TEXT'` data. Note that if component includes a private component resource holding this MIME data, it can use `GetComponentResource` to retrieve it. If the resource is a public component resource, it either use `GetComponentPublicResource` with the public type and ID or `GetComponentResource` with the private type and ID.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`QuickTimeComponents.h`

MovieImportGetDestinationMediaType

Returns the current type of a movie importer's destination media.

```
ComponentResult MovieImportGetDestinationMediaType (
   MovieImportComponent ci,
   OSType *mediaType
);
```

Parameters

ci

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

mediaType

A pointer to a media data type; see `Data References`.

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
`QuickTimeComponents.h`

MovieImportGetDontBlock

Undocumented

```
ComponentResult MovieImportGetDontBlock (
   MovieImportComponent ci,
   Boolean *willBlock
);
```

Parameters

ci

A movie data import component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

willBlock

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
`QuickTimeComponents.h`

MovieImportGetFileType

Allows your movie data import component to tell the Movie Toolbox the appropriate file type for the most-recently imported movie file.

```
ComponentResult MovieImportGetFileType (
   MovieImportComponent ci,
   OSType *fileType
);
```

Parameters

ci

A movie data import component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

fileType

A pointer to an `OSType` field. Your component should place the file type value that best identifies the movie data just imported. For example, Apple's Audio CD movie data import component sets this field to `'AIFF'` whenever it creates an AIFF file instead of a movie file.

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

Discussion
You should implement this function only if your movie data import component creates files other than QuickTime movies. By default, the Movie Toolbox makes new files into movies, unless you override that default by providing this function.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`QuickTimeComponents.h`

MovieImportGetLoadState

Undocumented

```
ComponentResult MovieImportGetLoadState (
   MovieImportComponent ci,
   long *importerLoadState
);
```

Parameters

ci

A movie data import component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

importerLoadState

Undocumented

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
`QuickTimeComponents.h`

MovieImportGetMaxLoadedTime

Undocumented

```
ComponentResult MovieImportGetMaxLoadedTime (
   MovieImportComponent ci,
   TimeValue *time
);
```

Parameters

ci

A movie data import component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

time

A pointer to a value containing the maximum loaded time.

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
`QuickTimeComponents.h`

MovieImportGetMIMETypeList

Returns a list of MIME types supported by the movie import component.

```
ComponentResult MovieImportGetMIMETypeList (
   MovieImportComponent ci,
   QTAtomContainer *mimeInfo
);
```

Parameters

ci

A movie data import component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

mimeInfo

A pointer to a MIME type list, a QT atom container that contains a list of MIME types supported by the movie import component. The caller should dispose of the atom container when finished with it.

Return Value

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

Discussion

Your movie import component can support MIME types that correspond to formats it supports. To make a list of these MIME types available to applications or other software, it must implement this function. To indicate that your movie import component supports this function, set the `hasMovieImportMIMEList` flag in the `componentFlags` field of the `ComponentDescription` structure.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`QuickTimeComponents.h`

MovieImportGetSampleDescription

Gets the current sample description for a movie import component.

```
ComponentResult MovieImportGetSampleDescription (
   MovieImportComponent ci,
   SampleDescriptionHandle *desc,
   OSType *mediaType
);
```

Parameters

ci

A movie data import component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

desc

A pointer to a handle to a `SampleDescription` structure.

mediaType

A pointer to the type of the data; see `Data References`.

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
`QuickTimeComponents.h`

MovieImportGetSettingsAsAtomContainer

Retrieves the current settings from the movie import component.

```
ComponentResult MovieImportGetSettingsAsAtomContainer (
   MovieImportComponent ci,
   QTAtomContainer *settings
);
```

Parameters

ci

A movie data import component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

settings

The address where the reference to the newly created atom container should be stored by the call.

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

Discussion
The caller is responsible for disposing of the returned QT atom container.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
Fiendishthngs

Declared In
`QuickTimeComponents.h`

MovieImportHandle

Imports data from a handle, using a movie data import component.

```
ComponentResult MovieImportHandle (
   MovieImportComponent ci,
   Handle dataH,
   Movie theMovie,
   Track targetTrack,
   Track *usedTrack,
   TimeValue atTime,
   TimeValue *addedDuration,
   long inFlags,
   long *outFlags
);
```

Parameters

ci

A movie data import component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

dataH

A handle to the data that is to be imported into the movie identified by the `theMovie` parameter. The data contained in this handle has a data type value that corresponds to your component's subtype value. Your component is not responsible for disposing of this handle.

theMovie

The movie for this operation. This movie identifier is supplied by the Movie Toolbox. Your component may use this identifier to add sample data to the target movie, or to obtain information about the movie.

targetTrack

The track that is to receive the imported data. This track identifier is supplied by the Movie Toolbox and is valid only if the `movieImportMustUseTrack` flag in the `inFlags` parameter is set to 1.

usedTrack

A pointer to the track that received the imported data. Your component returns this track identifier to the Movie Toolbox. Your component needs to set this parameter only if you operate on a single track or if you create a new track. If you modify more than one track, leave the field referred to by this parameter unchanged.

atTime

The time corresponding to the location where your component is to place the imported data. This time value is expressed in the movie's time coordinate system.

addedDuration

A pointer to the duration of the data that your component added to the movie. Your component must specify this value in the movie's time coordinate system.

inFlags

Flags (see below) that specify control information governing the import operation. See these constants:

`movieImportCreateTrack`

`movieImportMustUseTrack`

`movieImportInParallel`

outFlags

Flags (see below) that receive status information about the import operation. Your component sets the appropriate flags in this field when the operation is complete. See these constants:

`movieImportResultUsedMultipleTracks`

`movieImportInParallel`

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

Discussion
Your component must be prepared to perform this function at any time. You should not expect that any of your component's configuration functions will be called first. If your component can accept data from a handle, be sure to set the `canMovieImportHandles` flag in your component's `componentFlags` field.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`QuickTimeComponents.h`

MovieImportIdle

Undocumented

```
ComponentResult MovieImportIdle (
   MovieImportComponent ci,
   long inFlags,
   long *outFlags
);
```

Parameters

ci
: A movie data import component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

inFlags
: *Undocumented*

outFlags
: *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
`QuickTimeComponents.h`

MovieImportSetAuxiliaryData

Provides additional data to a component.

```
ComponentResult MovieImportSetAuxiliaryData (
   MovieImportComponent ci,
   Handle data,
   OSType handleType
);
```

Parameters

ci

A movie data import component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

data

A handle to the additional data. Your component should not dispose of this handle. Be sure to copy any data you need to keep.

handleType

The type of data in the specified handle.

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

Discussion
Your component should expect the application to call this function before the import process begins.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`QuickTimeComponents.h`

MovieImportSetChunkSize

The amount of data a component works with at a time.

```
ComponentResult MovieImportSetChunkSize (
   MovieImportComponent ci,
   long chunkSize
);
```

Parameters

ci

A movie data import component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

chunkSize

The number of seconds of data your movie data import component places into each chunk of movie data. This parameter may not be set to a value less than 1.

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

Discussion
Generally, your component should determine a reasonable default chunk size, based on the type of data you are importing. However, you may choose to allow applications to override your default value. This can be especially useful for sound data, where the chunk size affects the quality of sound playback.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`QuickTimeComponents.h`

MovieImportSetDimensions

Specifies a new track's spatial dimensions.

```
ComponentResult MovieImportSetDimensions (
   MovieImportComponent ci,
   Fixed width,
   Fixed height
);
```

Parameters

ci

A movie data import component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

width

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

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
`QuickTimeComponents.h`

MovieImportSetDontBlock

Undocumented

```
ComponentResult MovieImportSetDontBlock (
   MovieImportComponent ci,
   Boolean dontBlock
);
```

Parameters

ci

A movie data import component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

dontBlock

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
`QuickTimeComponents.h`

MovieImportSetDuration

Controls the duration of the data that a component pastes into the target movie.

```
ComponentResult MovieImportSetDuration (
   MovieImportComponent ci,
   TimeValue duration
);
```

Parameters

ci

A movie data import component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

duration

The duration in the movie's time scale. If this parameter is set to 0, then you may paste any amount of movie data that is appropriate for the data to be imported.

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

Discussion
If your component supports paste operations (that is, your component allows the application to set the `movieImportInParallel` flag to 1 with the `MovieImportHandle` (page 57) or `MovieImportFile` (page 50) function), then you must support this function. If an application calls this function and sets a duration limit, you must abide by that limit. This function is not valid for insert operations (where the `movieImportInParallel` flag is set to 0).

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`QuickTimeComponents.h`

MovieImportSetFromScrap

Indicates that the source data resides on the scrap.

```
ComponentResult MovieImportSetFromScrap (
   MovieImportComponent ci,
   Boolean fromScrap
);
```

Parameters

ci

A movie data import component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

fromScrap

Set to TRUE if the data originated on the scrap; otherwise, set to FALSE.

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
`QuickTimeComponents.h`

MovieImportSetIdleManager

Lets a movie importer report its idling needs.

```
ComponentResult MovieImportSetIdleManager (
   MovieImportComponent ci,
   IdleManager im
);
```

Parameters

ci

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

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 movie importer if it needs to report its idling requirements. In general, however, movie importers don't get idled. Typically, a movie importer just examines a file, scans it, and then determines if it can create a movie that will point at the file and describe how to play it. The media data is in that file, but the movie itself is in memory.

An idling importer is mostly used when you open a URL. For example, if you open an `.avi` file, the movie isn't completely constructed until the entire `.avi` file is downloaded. The job of the importer is to construct the movie, so the importer isn't going to be done constructing the movie until it is downloaded, which means you can't fast start an AVI movie. So the AVI importer returns immediately with a movie that is partially constructed. Every time QuickTime gets tasked, it gets some more time, but you can go ahead and start playing because it has already returned a movie, though one that is not complete yet.

An idling importer can return even before there's enough downloaded to construct a movie. It just creates an empty movie with no tracks and keep idling it, and eventually a movie appears.

Version Notes
Introduced in QuickTime 6.

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

Declared In
`QuickTimeComponents.h`

MovieImportSetMediaDataRef

Specifies a storage location that is to receive imported movie data.

```
ComponentResult MovieImportSetMediaDataRef (
   MovieImportComponent ci,
   Handle dataRef,
   OSType dataRefType
);
```

Parameters

ci
The component instance that identifies your connection to the graphics importer component.

dataRef
A data reference that specifies a storage location that receives the imported data.

dataRefType
The type of the data reference.

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

Discussion
By calling this function you can specify a storage location that receives some imported movie data.

Version Notes
Introduced in QuickTime 6.4.

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

Declared In
QuickTimeComponents.h

MovieImportSetMediaFile

Specifies a media file that is to receive the imported movie data.

```
ComponentResult MovieImportSetMediaFile (
   MovieImportComponent ci,
   AliasHandle alias
);
```

Parameters

ci

A movie data import component instance. Your software obtains this reference from OpenComponent or OpenDefaultComponent.

alias

The media file that is to receive the imported movie data. Your component must make a copy of this parameter. You should not dispose of it.

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
QuickTimeComponents.h

MovieImportSetNewMovieFlags

Implemented by a movie import component to determine the original flags for NewMovieFromDataRef.

```
ComponentResult MovieImportSetNewMovieFlags (
   MovieImportComponent ci,
   long newMovieFlags
);
```

Parameters

ci

A movie import component instance. Your software obtains this reference from OpenComponent or OpenDefaultComponent.

newMovieFlags

Constants (see below) that control characteristics of the new movie. See these constants:

newMovieActive

newMovieDontResolveDataRefs

newMovieDontAskUnresolvedDataRefs

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
`QuickTimeComponents.h`

MovieImportSetOffsetAndLimit

Specifies location and size of data that should be imported.

```
ComponentResult MovieImportSetOffsetAndLimit (
   MovieImportComponent ci,
   unsigned long offset,
   unsigned long limit
);
```

Parameters

ci

A movie data import component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

offset

A byte offset into the file that indicates where the import operation begins.

limit

A byte offset into the file that indicates the last data in the file that can be imported.

Return Value
See `Error Codes`. Returns `badComponentSelector` if the movie import component does not support this function. Returns `noErr` if there is no error.

Discussion
Typically, this function is used when the data is from a part of a file rather than the entire file. It is especially useful when one file format is embedded in another; it allows your application to skip header data for the enclosing file and begin importing data at the start of the desired format.

Special Considerations

Not all movie import components support this function. Those that do include the movie import components for the `kQTFileTypeAIFF`, `kQTFileTypeWave`, and `kQTFileTypeMuLaw` file types. Those that do not return the `badComponentSelector` result code in response to a this call.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`QuickTimeComponents.h`

MovieImportSetOffsetAndLimit64

Specifies location and size of data that should be imported from a file.

```
ComponentResult MovieImportSetOffsetAndLimit64 (
   MovieImportComponent ci,
   const wide *offset,
   const wide *limit
);
```

Parameters

ci

A movie data import component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

offset

A byte offset into the file that indicates where the import operation begins.

limit

A byte offset into the file that indicates the last data in the file that can be imported.

Return Value

See `Error Codes`. Returns `badComponentSelector` if the movie import component does not support this function. Returns `noErr` if there is no error.

Discussion

This function serves the same purpose as `MovieImportSetOffsetAndLimit` (page 66). The only difference is that the offset and limit can hold 64-bit offsets. This function is especially useful when one file format is embedded in another; it allows your application to skip header data for the enclosing file and begin importing data at the start of the desired format.

Special Considerations

Not all movie import components support this function. Those that do not return the `badComponentSelector` result code. If this function is not implemented and the offset and limit can be expressed using 32-bit offsets, `MovieImportSetOffsetAndLimit` (page 66) should be tried.

Version Notes

Introduced in QuickTime 4.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`QuickTimeComponents.h`

MovieImportSetProgressProc

Assigns a movie progress function.

```
ComponentResult MovieImportSetProgressProc (
   MovieImportComponent ci,
   MovieProgressUPP proc,
   long refcon
);
```

Parameters

ci

A movie data import component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

proc

A pointer to the application's `MovieProgressProc` callback. If this parameter is set to `NIL`, the application is removing its progress function. In this case, your component should stop calling the progress function.

refcon

Specifies a reference constant. Your component should pass this constant back to the application's progress function whenever you call that function. The application may use this parameter to point to a data structure containing any information the callback needs.

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

Discussion
The `MovieProgressProc` callback interface not only allows you to report progress to the application, but also allows the application to cancel the request.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`QuickTimeComponents.h`

MovieImportSetSampleDescription

Provides a SampleDescription structure to a movie data import component.

```
ComponentResult MovieImportSetSampleDescription (
   MovieImportComponent ci,
   SampleDescriptionHandle desc,
   OSType mediaType
);
```

Parameters

ci

A movie data import component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

desc

A handle to a `SampleDescription` structure. Your component must not dispose of this handle. If you want to save any data from the structure, be sure to copy it at this time.

mediaType

The type of sample description referred to by the `desc` parameter. If the `desc` parameter refers to an `ImageDescription` structure, this parameter is set to `VideoMediaType` (`'vide'`); for `SoundDescription` structures, this parameter is set to `SoundMediaType` (`'soun'`).

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
`QuickTimeComponents.h`

MovieImportSetSampleDuration

Sets the sample duration for new samples to be created with a component.

```
ComponentResult MovieImportSetSampleDuration (
   MovieImportComponent ci,
   TimeValue duration,
   TimeScale scale
);
```

Parameters

ci

A movie data import component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

duration

The sample duration in units specified by the `scale` parameter.

scale

The time scale for the duration value. This may be any arbitrary time scale; that is, it may not correspond to the movie's time scale. You should convert this time scale to the movie's time scale before using the duration value, using `ConvertTimeScale`.

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
`QuickTimeComponents.h`

MovieImportSetSettingsFromAtomContainer

Sets the movie import component's current configuration from the passed settings data.

```
ComponentResult MovieImportSetSettingsFromAtomContainer (
   MovieImportComponent ci,
   QTAtomContainer settings
);
```

Parameters

ci

A movie data import component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

settings

A QT atom container containing settings.

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

Discussion
The settings QT atom container may contain atoms other than those expected by the particular component type or may be missing certain atoms. The function uses only those settings it understands.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`QuickTimeComponents.h`

MovieImportValidate

Allows your movie data import component to validate the data to be passed to your component.

```
ComponentResult MovieImportValidate (
   MovieImportComponent ci,
   const FSSpec *theFile,
   Handle theData,
   Boolean *valid
);
```

Parameters

ci

A movie data import component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

theFile

An `FSSpec` structure that defines the file to validate if the importer imports from files.

theData

The data to validate if the importer imports from handles.

valid

A pointer to a Boolean value. If the data or file can be imported, the value returned is TRUE. Otherwise, it returns FALSE.

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

Discussion
Movie import components can implement this function to allow applications to determine if a given file or handle to data is acceptable for a particular import component. As this function may be called on many files, the validation process should be as fast as possible.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`QuickTimeComponents.h`

MovieImportValidateDataRef

Validates the data file indicated by the data reference.

```
ComponentResult MovieImportValidateDataRef (
   MovieImportComponent ci,
   Handle dataRef,
   OSType dataRefType,
   UInt8 *valid
);
```

Parameters

ci

A movie data import component instance. Your software obtains this reference from `OpenComponent` or `OpenDefaultComponent`.

dataRef

The data reference to the file to be validated.

dataRefType

The type of data reference for the `dataRef` parameter.

valid

A pointer to a UInt8 value. If the data or file cannot be imported, the value returned should be 0. Otherwise, it should be set to 128.

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

Discussion
Movie import components can implement this function to allow applications to determine if a given file referenced by a data reference is acceptable for a particular import component. The data reference can refer to any data for which there is a suitable data handler component installed and available to QuickTime. As this function may be called on many files, the validation process should be as fast as possible. Furthermore, the importer should probably limit the amount of reading it performs, especially when the data handler refers to data on the Internet.

Special Considerations

Unlike `MovieImportValidate` (page 70), the `valid` parameter for this function is a value that can be interpreted as the degree to which the importer can interpret the file's contents. In all cases, returning 0 indicates the file cannot be interpreted by the importer. However, other non-zero values can be used to determine the relative weighting between multiple importers that can import a particular kind of file. For now, it is best to return either 0 or 128 only.

Version Notes
Introduced in QuickTime 4.

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

Related Sample Code
ElectricImageComponent
ElectricImageComponent.win

Declared In
`QuickTimeComponents.h`

NewMovieExportGetDataUPP

Allocates a Universal Procedure Pointer for the MovieExportGetDataProc callback.

```
MovieExportGetDataUPP NewMovieExportGetDataUPP (
   MovieExportGetDataProcPtr 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 `NewMovieExportGetDataProc`.

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

Related Sample Code
CIVideoDemoGL
qtmoviefromprocs
qtmoviefromprocs.win

Declared In
`QuickTimeComponents.h`

NewMovieExportGetPropertyUPP

Allocates a Universal Procedure Pointer for the MovieExportGetPropertyProc callback.

```
MovieExportGetPropertyUPP NewMovieExportGetPropertyUPP (
   MovieExportGetPropertyProcPtr 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 `NewMovieExportGetPropertyProc`.

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

Related Sample Code
CIVideoDemoGL
qtmoviefromprocs
qtmoviefromprocs.win

Declared In
`QuickTimeComponents.h`

NewMovieExportStageReachedCallbackUPP

Allocates a new Universal Procedure Pointer for a MovieExportStageReachedCallbackProc callback.

```
MovieExportStageReachedCallbackUPP NewMovieExportStageReachedCallbackUPP (
   MovieExportStageReachedCallbackProcPtr userRoutine
);
```

Parameters

userRoutine

A pointer to your application-defined callback function; see `ICMDecompressionTrackingCallbackProc`.

Return Value
A new Universal Procedure Pointer that you will use to invoke your callback.

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

Declared In
`QuickTimeComponents.h`

NewSCModalFilterUPP

Allocates a Universal Procedure Pointer for the SCModalFilterProc callback.

```
SCModalFilterUPP NewSCModalFilterUPP (
   SCModalFilterProcPtr 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 `NewSCModalFilterProc`.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

qtcompress

qtcompress.win

Declared In

`QuickTimeComponents.h`

NewSCModalHookUPP

Allocates a Universal Procedure Pointer for the SCModalHookProc callback.

```
SCModalHookUPP NewSCModalHookUPP (
   SCModalHookProcPtr 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 `NewSCModalHookProc`.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code
ConvertToMovieJr
qtcompress
qtcompress.win
VideoProcessing

Declared In
`QuickTimeComponents.h`

PreviewEvent

May be called as appropriate if a preview component handles events.

```
ComponentResult PreviewEvent (
   pnotComponent p,
   EventRecord *e,
   Boolean *handledEvent
);
```

Parameters

p
Specifies your preview component. You obtain this identifier from `OpenComponent`.

e
A pointer to the event structure for this operation.

handledEvent
A pointer to a Boolean value. If you completely handle an event such as a mouse-down event or keystroke, you should set the `handledEvent` parameter to TRUE. Otherwise, set it to FALSE.

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
`QuickTimeComponents.h`

PreviewMakePreview

Creates previews by allocating a handle to data that is to be added to a file.

```
ComponentResult PreviewMakePreview (
   pnotComponent p,
   OSType *previewType,
   Handle *previewResult,
   const FSSpec *sourceFile,
   ICMProgressProcRecordPtr progress
);
```

Parameters

p

Specifies your preview component. You obtain this identifier from `OpenComponent`.

previewType

A pointer to the type of preview component that should be used to display the preview.

previewResult

A pointer to a handle of cached preview data created by this function.

sourceFile

A pointer to a reference to the file for which the preview is created.

progress

A pointer to an `ICMProgressProcRecord` 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
`QuickTimeComponents.h`

PreviewMakePreviewReference

Returns the type and identification number of a resource within a file to be used as the preview for a file.

```
ComponentResult PreviewMakePreviewReference (
   pnotComponent p,
   OSType *previewType,
   short *resID,
   const FSSpec *sourceFile
);
```

Parameters

p

Specifies your preview component. You obtain this identifier from `OpenComponent`.

previewType

A pointer to the type of preview component that should be used to display the preview.

resID

A pointer to the identification number of a resource within the file to be used as the preview for the file.

sourceFile

A pointer to an `FSSpec` structure that provides a reference to the file for which the preview is created.

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
`QuickTimeComponents.h`

PreviewShowData

Displays a preview if it does not handle events.

```
ComponentResult PreviewShowData (
   pnotComponent p,
   OSType dataType,
   Handle data,
   const Rect *inHere
);
```

Parameters

p

Specifies your preview component. You obtain this identifier from `OpenComponent`.

dataType

The type of handle pointing to the data to be displayed in the preview.

data

A handle to the data, which is typically the same as the subtype of your preview component.

inHere

A pointer to a `Rect` structure that defines the area into which you draw the preview. The current port is set to the correct graphics port for drawing. You must not draw outside the given rectangle.

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
AlwaysPreview

Declared In
`QuickTimeComponents.h`

SCAsyncIdle

Called occasionally while performing asynchronous compression with SCCompressSequenceFrameAsync.

```
ComponentResult SCAsyncIdle (
   ComponentInstance ci
);
```

Parameters

ci

Your application's connection to the image-compression component being used by `SCCompressSequenceFrameAsync` (page 84). You obtain this identifier from `OpenDefaultComponent`.

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

qtcompress

qtcompress.win

Declared In

`QuickTimeComponents.h`

SCAudioInvokeLegacyCodecOptionsDialog

Invokes the legacy code options dialog of an audio codec component.

```
ComponentResult SCAudioInvokeLegacyCodecOptionsDialog (
   ComponentInstance ci
);
```

Parameters

ci

A component instance that identifies a connection to an audio codec component.

Return Value

An error code, or `noErr` if there is no error.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`QuickTimeComponents.h`

SCCompressImage

Compresses an image that is stored in a PixMap structure.

```
ComponentResult SCCompressImage (
   ComponentInstance ci,
   PixMapHandle src,
   const Rect *srcRect,
   ImageDescriptionHandle *desc,
   Handle *data
);
```

Parameters

ci

Identifies your application's connection to a standard image-compression dialog component. You obtain this identifier from `OpenDefaultComponent`.

src

A handle to the `PixMap` structure to be compressed.

srcRect

A pointer to a portion of the `PixMap` structure to compress as a `Rect` structure. This rectangle must be in the pixel map's coordinate system. If you want to compress the entire pixel map, set this parameter to `NIL`.

desc

A pointer to a handle to an `ImageDescription` structure. The standard dialog component creates an `ImageDescription` structure when it compresses the image, and returns a handle to that structure in the field referred to by this parameter. The component sizes that handle appropriately. Your application is responsible for disposing of that handle when you are done with it.

data

A pointer to a handle. The standard dialog component returns a handle to the compressed image data in the field referred to by this parameter. The component sizes that handle appropriately. Your application is responsible for disposing of that handle when you are done with it.

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
qtcompress
qtcompress.win
qtstdcompr
qtstdcompr.win

Declared In
`QuickTimeComponents.h`

SCCompressPicture

Compresses a Picture structure that is stored by a handle.

```
ComponentResult SCCompressPicture (
   ComponentInstance ci,
   PicHandle srcPicture,
   PicHandle dstPicture
);
```

Parameters

ci

Identifies your application's connection to a standard image-compression dialog component. You obtain this identifier from `OpenDefaultComponent`.

srcPicture

A handle to the `Picture` structure to be compressed.

dstPicture

A handle to the compressed `Picture` structure. The standard dialog component resizes this handle to accommodate the compressed structure. Your application is responsible for creating and disposing of this handle when you are done with it.

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
`QuickTimeComponents.h`

SCCompressPictureFile

Compresses a Picture structure that is stored in a file.

```
ComponentResult SCCompressPictureFile (
   ComponentInstance ci,
   short srcRefNum,
   short dstRefNum
);
```

Parameters

ci

Identifies your application's connection to a standard image-compression dialog component. You obtain this identifier from `OpenDefaultComponent`.

srcRefNum

A reference to the file to be compressed.

dstRefNum

A reference to the file that is to receive the compressed data. This may be the same as the source file. The standard dialog component places the compressed image data into the file identified by this reference. Your application is responsible for this file after the compression 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.

Related Sample Code
Std Compression Examples

Declared In
`QuickTimeComponents.h`

SCCompressSequenceBegin

Initiates a sequence-compression operation.

```
ComponentResult SCCompressSequenceBegin (
   ComponentInstance ci,
   PixMapHandle src,
   const Rect *srcRect,
   ImageDescriptionHandle *desc
);
```

Parameters

ci

Identifies your application's connection to a standard image-compression component. You obtain this identifier from `OpenDefaultComponent`.

src

A handle to the `PixMap` structure to be compressed. This pixel map must contain the first image in the sequence.

srcRect

A pointer to a portion of the `PixMap` structure to compress as a `Rect` structure. This rectangle must be in the pixel map's coordinate system. If you want to compress the entire structure, set this parameter to `NIL`.

desc

A pointer to an image description handle. The standard dialog component creates an image description structure when it compresses the image, and returns a handle to that structure in the field referred to by this parameter. The component sizes the handle appropriately. If you do not want this information, set this parameter to `NIL`.

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
CompressMovies
ConvertToMovieJr
qtcompress

qtcompress.win
VideoProcessing

Declared In
`QuickTimeComponents.h`

SCCompressSequenceEnd

Ends a sequence-compression operation.

```
ComponentResult SCCompressSequenceEnd (
   ComponentInstance ci
);
```

Parameters

ci

Identifies your application's connection to a standard image-compression component. You obtain this identifier from `OpenDefaultComponent`.

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

Discussion
The standard dialog component disposes of any memory it used to compress the image sequence, including the data and image description buffers. You must call this function once for each sequence you start.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
CompressMovies
ConvertToMovieJr
qtcompress
qtcompress.win
VideoProcessing

Declared In
`QuickTimeComponents.h`

SCCompressSequenceFrame

Continues a sequence-compression operation.

```
ComponentResult SCCompressSequenceFrame (
   ComponentInstance ci,
   PixMapHandle src,
   const Rect *srcRect,
   Handle *data,
   long *dataSize,
   short *notSyncFlag
);
```

Parameters

ci

Identifies your application's connection to a standard image-compression component. You obtain this identifier from `OpenDefaultComponent`.

src

A handle to the `PixMap` structure to be compressed.

srcRect

A pointer to a portion of the `PixMap` structure to compress as a `Rect` structure. This rectangle must be in the pixel map's coordinate system. If you want to compress the entire pixel map, set this parameter to `NIL`.

data

A pointer to a handle. The standard compression component returns a handle to the compressed image data in the field referred to by this parameter. The component sizes that handle appropriately for the sequence.

dataSize

A pointer to a long integer. The standard compression component returns a value that indicates the number of bytes of compressed image data that it returns. Note that this value will differ from the size of the handle referred to by the `data` parameter, because the handle is allocated to accommodate the largest image in the sequence.

notSyncFlag

A pointer to a short integer that indicates whether the compressed frame is a key frame. If the frame is a key frame, the standard compression component sets the field referred to by this parameter to 0; otherwise, the `component` sets this field to `mediaSampleNotSync`. You may use this field to set the `sampleFlags` parameter of the `AddMediaSample` function.

Return Value

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

Discussion

You must call this function once for each frame in the sequence, including the first frame.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

CompressMovies

ConvertToMovieJr

qtcompress

qtcompress.win

VideoProcessing

Declared In
QuickTimeComponents.h

SCCompressSequenceFrameAsync

An asynchronous variant of SCCompressSequenceFrame, with a completion callback.

```
ComponentResult SCCompressSequenceFrameAsync (
   ComponentInstance ci,
   PixMapHandle src,
   const Rect *srcRect,
   Handle *data,
   long *dataSize,
   short *notSyncFlag,
   ICMCompletionProcRecordPtr asyncCompletionProc
);
```

Parameters

ci

Identifies your application's connection to a standard image-compression component. You obtain this identifier from `OpenDefaultComponent`.

src

A handle to the `PixMap` structure to be compressed.

srcRect

A pointer to a portion of the `PixMap` structure to compress as a `Rect` structure. This rectangle must be in the pixel map's coordinate system. If you want to compress the entire pixel map, set this parameter to `NIL`.

data

A pointer to a handle. The standard compression component returns a handle to the compressed image data in the field referred to by this parameter. The component sizes that handle appropriately for the sequence.

dataSize

A pointer to a long integer. The standard compression component returns a value that indicates the number of bytes of compressed image data that it returns. Note that this value will differ from the size of the handle referred to by the `data` parameter, because the handle is allocated to accommodate the largest image in the sequence.

notSyncFlag

A pointer to a short integer that indicates whether the compressed frame is a key frame. If the frame is a key frame, the standard compression component sets the field referred to by this parameter to 0; otherwise, the `component` sets this field to `mediaSampleNotSync`. You may use this field to set the `sampleFlags` parameter of the `AddMediaSample` function.

asyncCompletionProc

A pointer to an `ICMCompletionProcRecord` structure. If you pass `NIL`, the `SCCompressSequenceFrameAsync` function acts like `SCCompressSequenceFrame` (page 82).

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

Discussion
While performing asynchronous compression with this function, you should occasionally call `SCAsyncIdle` (page 78). This gives the standard compression component an opportunity to restart its compression operation if it needs to force a key frame.

Version Notes
Introduced in QuickTime 5.

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

Related Sample Code
qtcompress
qtcompress.win

Declared In
`QuickTimeComponents.h`

SCCopyCompressionSessionOptions

Creates a compression session options object based upon the settings in the Standard Compression component.

```
ComponentResult SCCopyCompressionSessionOptions (
   ComponentInstance ci,
   ICMCompressionSessionOptionsRef *outOptions
);
```

Parameters

ci
A component instance of Standard Compression component.

outOptions
On return, a reference to a new compression session options object.

Return Value
An error code. Returns `noErr` if there is no error. `paramErr` if the client did not set the `scAllowEncodingWithCompressionSession` preference flag.

Discussion
This function creates a new compression session options object using the compression settings of the Standard Compression component instance. You can use other Standard Compression component calls to set up the compression settings. Then you call this function to extract the compression settings in the form of a compression session options object. The returned object can be used to create a compression session object through `ICMCompressionSessionCreate`().

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

Related Sample Code
OpenGLCaptureToMovie
Quartz Composer QCTV

Declared In
`QuickTimeComponents.h`

SCDefaultPictFileSettings

Derives default compression settings for a Picture structure that is stored in a file.

```
ComponentResult SCDefaultPictFileSettings (
   ComponentInstance ci,
   short srcRef,
   short motion
);
```

Parameters

ci

Identifies your application's connection to a standard image-compression dialog component. You obtain this identifier from `OpenDefaultComponent`.

srcRef

A reference to the file to be analyzed.

motion

Specifies whether the image is part of a sequence. Set this parameter to TRUE if the image is part of a sequence; set it to FALSE if you are working with a single still image.

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

Std Compression Examples

Declared In

`QuickTimeComponents.h`

SCDefaultPictHandleSettings

Derives default compression settings for a Picture structure that is stored by a handle.

```
ComponentResult SCDefaultPictHandleSettings (
   ComponentInstance ci,
   PicHandle srcPicture,
   short motion
);
```

Parameters

ci

Identifies your application's connection to a standard image-compression dialog component. You obtain this identifier from `OpenDefaultComponent`.

srcPicture

A handle to the `Picture` structure to be analyzed.

motion

Specifies whether the image is part of a sequence. Set this parameter to TRUE if the image is part of a sequence; set it to FALSE if you are working with a single still image.

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
`QuickTimeComponents.h`

SCDefaultPixMapSettings

Derives default compression settings for an image that is stored in a pixel map.

```
ComponentResult SCDefaultPixMapSettings (
   ComponentInstance ci,
   PixMapHandle src,
   short motion
);
```

Parameters

ci

Identifies your application's connection to a standard image-compression dialog component. You obtain this identifier from `OpenDefaultComponent`.

src

A handle to the `PixMap` structure to be analyzed.

motion

Specifies whether the image is part of a sequence. Set this parameter to TRUE if the image is part of a sequence; set it to FALSE if you are working with a single still image.

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
CompressMovies
ConvertToMovieJr
qtcompress
qtcompress.win
VideoProcessing

Declared In
`QuickTimeComponents.h`

SCGetBestDeviceRect

Determines the boundary rectangle that surrounds the display device that supports the largest color or grayscale palette.

```
ComponentResult SCGetBestDeviceRect (
   ComponentInstance ci,
   Rect *r
);
```

Parameters

ci

Identifies your application's connection to a standard image-compression dialog component. You obtain this identifier from `OpenDefaultComponent`.

r

A pointer to a `Rect` structure. The function returns the global coordinates of a rectangle that surrounds the appropriate display device.

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

Discussion
The standard image-compression dialog component uses this function to position rectangles and dialog boxes when you indicate that the component is to choose the best display device. It subtracts the menu bar from the returned rectangle if the best device is also the main display device.

Special Considerations

In general, your application does not need to use this function.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`QuickTimeComponents.h`

SCGetCompressFlags

Gets compression flags for a standard image-compression dialog component.

```
ComponentResult SCGetCompressFlags (
   ComponentInstance ci,
   long *flags
);
```

Parameters

ci

Identifies your application's connection to a standard image-compression dialog component. You obtain this identifier from `OpenDefaultComponent`.

flags

A pointer to compression flags (see below). See these constants:
`scCompressFlagIgnoreIdenticalFrames`

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
`QuickTimeComponents.h`

SCGetCompressionExtended

Undocumented

```
ComponentResult SCGetCompressionExtended (
   ComponentInstance ci,
   SCParams *params,
   Point where,
   SCModalFilterUPP filterProc,
   SCModalHookUPP hookProc,
   long refcon,
   StringPtr customName
);
```

Parameters

ci
: Identifies your application's connection to a standard image-compression dialog component. You obtain this identifier from `OpenDefaultComponent`.

params
: A pointer to an `SCParams` structure.

where
: *Undocumented*

filterProc
: A Universal Procedure Pointer that accesses a `SCModalFilterProc` callback.

hookProc
: A Universal Procedure Pointer that accesses a `SCModalHookProc` callback.

refcon
: A reference constant to be passed to your callbacks. Use this parameter to point to a data structure containing any information your callbacks need.

customName
: *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
QuickTimeComponents.h

SCGetInfo

Retrieves configuration information from the standard dialog component.

```
ComponentResult SCGetInfo (
   ComponentInstance ci,
   OSType infoType,
   void *info
);
```

Parameters

ci

Identifies your application's connection to a standard image-compression dialog component. You obtain this identifier from `OpenDefaultComponent`.

infoType

A constant (see below) that specifies the type of information you want to retrieve. See these constants:

`scSpatialSettingsType`
`scTemporalSettingsType`
`scDataRateSettingsType`
`scColorTableType`
`scProgressProcType`
`scExtendedProcsType`
`scPreferenceFlagsType`
`scSettingsStateType`
`scSequenceIDType`
`scWindowPositionType`
`scCodecFlagsType`

info

A pointer to a field that is to receive the information. The `infoType` constant descriptions (see below) include information about this field.

Return Value
See `Error Codes`. If the component cannot satisfy your request, it returns a result code of `scTypeNotFoundErr`. It 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
CompressMovies
ConvertMovieSndTrack
ConvertToMovieJr
qtcompress.win
VideoProcessing

Declared In
`QuickTimeComponents.h`

SCGetSettingsAsAtomContainer

Places the current configuration from the standard image-compression component in a QT atom container.

```
ComponentResult SCGetSettingsAsAtomContainer (
   ComponentInstance ci,
   QTAtomContainer *settings
);
```

Parameters

ci

The standard compression component instance.

settings

The address where the newly-created atom container should be stored.

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

Discussion
The caller is responsible for disposing of the returned QT atom container.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
ElectricImageComponent
ElectricImageComponent.win
Quartz Composer QCTV
ThreadsExportMovie
vrmakepano

Declared In
`QuickTimeComponents.h`

SCGetSettingsAsText

Undocumented

```
ComponentResult SCGetSettingsAsText (
   ComponentInstance ci,
   Handle *text
);
```

Parameters

ci

Identifies your application's connection to a standard image-compression dialog component. You obtain this identifier from `OpenDefaultComponent`.

text

A pointer to a handle to text.

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
`QuickTimeComponents.h`

SCNewGWorld

Creates a graphics world based on the current compression settings.

```
ComponentResult SCNewGWorld (
   ComponentInstance ci,
   GWorldPtr *gwp,
   Rect *rp,
   GWorldFlags flags
);
```

Parameters

ci

Identifies your application's connection to a standard image-compression dialog component. You obtain this identifier from `OpenDefaultComponent`.

gwp

A pointer to a pointer to a `CGrafPort` structure that defines a graphics world. The standard dialog component places a pointer to the new graphics world into the field referred to by this parameter. If the component cannot create the graphics world, it sets this field to `NIL`.

rp

A pointer to the boundaries of the graphics world. If you set this parameter to `NIL`, the standard dialog component uses the test image's boundary rectangle. If you don't specify a boundary rectangle and there is no test image, the component does not create the graphics world.

flags

Contains flags (see below) that determine some of the memory characteristics of the new graphics world. See these constants:

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
`QuickTimeComponents.h`

SCPositionDialog

Helps position a dialog box on the screen.

```
ComponentResult SCPositionDialog (
   ComponentInstance ci,
   short id,
   Point *where
);
```

Parameters

ci

Identifies your application's connection to a standard image-compression dialog component. You obtain this identifier from `OpenDefaultComponent`.

id

The resource number of a `'DLOG'` resource. The function positions the dialog box that corresponds to this resource.

where

A pointer to a `Point` structure identifying the desired location of the upper-left corner of the dialog box in global coordinates. This parameter allows you to indicate how you want to position the dialog box on the screen.

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
ConvertToMovieJr

VideoProcessing

Declared In
`QuickTimeComponents.h`

SCPositionRect

Positions a rectangle on the screen.

```
ComponentResult SCPositionRect (
   ComponentInstance ci,
   Rect *rp,
   Point *where
);
```

Parameters

ci

Identifies your application's connection to a standard image-compression dialog component. You obtain this identifier from `OpenDefaultComponent`.

rp

A pointer to a `Rect` structure. When you call the function, this structure should contain the rectangle's current global coordinates. The function adjusts the coordinates in the structure to reflect the rectangle's new position.

where

A pointer to a `Point` structure identifying the desired location of the upper-left corner of the rectangle in global coordinates. This parameter allows your application to position the rectangle on the screen.

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
CompressMovies
ConvertToMovieJr
VideoProcessing

Declared In
`QuickTimeComponents.h`

SCRequestImageSettings

Displays the standard image dialog box to the user and shows default settings you have established.

```
ComponentResult SCRequestImageSettings (
   ComponentInstance ci
);
```

Parameters

ci

Identifies your application's connection to a standard image-compression dialog component. You obtain this identifier from `OpenDefaultComponent`.

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

Discussion
Use this function to retrieve the user's preferences for compressing a single image; use `SCRequestSequenceSettings` (page 95) when you are working with an image sequence. Both functions manipulate the compression settings that the component stores for you.

The component derives the current settings when you may supply an image to the component from which it can derive default settings. If you have not set any defaults, but you do supply a test image for the dialog, the component examines the test image and derives appropriate default values based upon its characteristics. If you have not set any defaults and do not supply a test image, the component uses its own default values.

Special Considerations

You may modify the settings by using `SCSetInfo` (page 96). You may customize the dialog boxes by specifying a modal-dialog hook function or a custom button. You may use the custom button to invoke an ancillary dialog box that is specific to your application.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
BurntTextSampleCode
QTExtractAndConvertToAIFF
qtstdcompr
soundsnippets.win
Std Compression Examples

Declared In
`QuickTimeComponents.h`

SCRequestSequenceSettings

Displays the standard sequence dialog box to the user and shows default settings you have established.

```
ComponentResult SCRequestSequenceSettings (
   ComponentInstance ci
);
```

Parameters

ci

Identifies your application's connection to a standard image-compression dialog component. You obtain this identifier from `OpenDefaultComponent`.

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

Discussion
Use `SCRequestSequenceSettings` to retrieve the user's preferences for compressing an image sequence; use `SCRequestImageSettings` (page 94) when you are working with a single image. Both functions manipulate the compression settings that the component stores for you.

The component derives the current settings when you may supply an image to the component from which it can derive default settings. If you have not set any defaults, but you do supply a test image for the dialog, the component examines the test image and derives appropriate default values based upon its characteristics. If you have not set any defaults and do not supply a test image, the component uses its own default values.

Special Considerations

You may modify the settings by using `SCSetInfo` (page 96). You may customize the dialog boxes by specifying a modal-dialog hook function or a custom button. You may use the custom button to invoke an ancillary dialog box that is specific to your application.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
CompressMovies
ConvertToMovieJr
OpenGLCaptureToMovie
qtcompress
Quartz Composer QCTV

Declared In
`QuickTimeComponents.h`

SCSetCompressFlags

Sets compression flags for a standard image-compression dialog component.

```
ComponentResult SCSetCompressFlags (
   ComponentInstance ci,
   long flags
);
```

Parameters

ci

Identifies your application's connection to a standard image-compression dialog component. You obtain this identifier from `OpenDefaultComponent`.

flags

Flags (see below) to set. See these constants:
`scCompressFlagIgnoreIdenticalFrames`

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
`QuickTimeComponents.h`

SCSetInfo

Modifies the standard dialog component's configuration information.

```
ComponentResult SCSetInfo (
   ComponentInstance ci,
   OSType infoType,
   void *info
);
```

Parameters

ci

Identifies your application's connection to a standard image-compression dialog component. You obtain this identifier from `OpenDefaultComponent`.

infoType

A constant (see below) that specifies the type of information you want to set. See these constants:

`scSpatialSettingsType`
`scTemporalSettingsType`
`scDataRateSettingsType`
`scColorTableType`
`scProgressProcType`
`scExtendedProcsType`
`scPreferenceFlagsType`
`scSettingsStateType`
`scSequenceIDType`
`scWindowPositionType`
`scCodecFlagsType`

info

A pointer to a field that contains the new information. The `infoType` constant descriptions (see below) include information about this field.

Return Value

See `Error Codes`. If the component cannot satisfy your request, it returns a result code of `scTypeNotFoundErr`. It 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

CompressMovies
ConvertToMovieJr
qtcompress
qtcompress.win
VideoProcessing

Declared In

`QuickTimeComponents.h`

SCSetSettingsFromAtomContainer

Sets the standard image-compression component's current configuration from data in a QT atom container.

```
ComponentResult SCSetSettingsFromAtomContainer (
   ComponentInstance ci,
   QTAtomContainer settings
);
```

Parameters

ci

Standard compression component instance.

settings

A QT atom container reference to the settings.

Return Value

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

Discussion

The settings QT atom container may contain atoms other than those expected by the particular component type or may be missing certain atoms. The function will only use settings it understands.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

ConvertMovieSndTrack

OpenGLCaptureToMovie

Quartz Composer QCTV

Declared In

`QuickTimeComponents.h`

SCSetTestImagePictFile

Sets the dialog box's test image from a Picture structure that is stored in a picture file.

```
ComponentResult SCSetTestImagePictFile (
   ComponentInstance ci,
   short testFileRef,
   Rect *testRect,
   short testFlags
);
```

Parameters

ci

Identifies your application's connection to a standard image-compression dialog component. You obtain this identifier from `OpenDefaultComponent`.

testFileRef

Identifies the file that contains the new test image. Your application is responsible for opening this file before calling this function. You must also close the file when you are done with it. You must clear the image or close your connection to the standard image-compression dialog component before you close the file. If the file contains a large image, the component may take some time to display the standard image-compression dialog box. In this case, the component displays the watch cursor while it loads the test image.

testRect

A pointer to a `Rect` structure. This rectangle specifies, in the coordinate system of the source image, the area of interest or point of interest in the test image. The area of interest defines a portion of the test image that is to be shown to the user in the dialog box. Use this parameter to direct the component to a specific portion of the test image. The component uses the value of the `testFlags` parameter to determine how it transforms large images before displaying them to the user.

testFlags

Constants (see below) that specify how the component is to display a test image that is larger than the test image portion of the dialog box. If you set this parameter to 0, the component uses a default method of its own choosing. In all cases, the component centers the area or point of interest in the test image portion of the dialog box, and then displays some part of the test image. See these constants:

`scPreferCropping`

`scPreferScaling`

`scPreferScalingAndCropping`

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
Std Compression Examples

Declared In
`QuickTimeComponents.h`

SCSetTestImagePictHandle

Sets the dialog box's test image from a Picture structure that is stored in a handle.

```
ComponentResult SCSetTestImagePictHandle (
   ComponentInstance ci,
   PicHandle testPict,
   Rect *testRect,
   short testFlags
);
```

Parameters

ci

Identifies your application's connection to a standard image-compression dialog component. You obtain this identifier from `OpenDefaultComponent`.

testPict

Identifies a handle that contains the new test image. Your application is responsible for disposing of this handle when you are done with it. You must clear the image or close your connection to the standard image-compression dialog component before you dispose of this handle or close the corresponding resource file. You must set this handle as nonpurgeable.

testRect

A pointer to a `Rect` structure. This structure specifies, in the coordinate system of the source image, the area of interest or point of interest in the test image. The area of interest defines a portion of the test image that is to be shown to the user in the dialog box. Use this parameter to direct the component to a specific portion of the test image. The component uses the value of the `testFlags` parameter to determine how it transforms this image before displaying it to the user. The component uses the `testFlags` parameter only when the test image is larger than the test image portion of the dialog box.

testFlags

Constants (see below) that specify how the component is to display a test image that is larger than the test image portion of the dialog box. If you set this parameter to 0, the component uses a default method of its own choosing. In all cases, the component centers the area or point of interest in the test image portion of the dialog box, and then displays some part of the test image. See these constants:

`scPreferCropping`

`scPreferScaling`

`scPreferScalingAndCropping`

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

`QuickTimeComponents.h`

SCSetTestImagePixMap

Sets the dialog box's test image from a Picture structure that is stored in a PixMap structure.

```
ComponentResult SCSetTestImagePixMap (
   ComponentInstance ci,
   PixMapHandle testPixMap,
   Rect *testRect,
   short testFlags
);
```

Parameters

ci

Identifies your application's connection to a standard image-compression dialog component. You obtain this identifier from `OpenDefaultComponent`.

testPixMap

A handle to a `PixMap` structure that contains the new test image. Your application is responsible for creating this structure before calling the function. You must also dispose of the structure when you are done with it. You must clear the image or close your connection to the standard image-compression dialog component before you dispose of the structure.

testRect

A pointer to a `Rect` structure. This rectangle specifies, in the coordinate system of the source image, the area of interest or point of interest in the test image. The area of interest defines a portion of the test image that is to be shown to the user in the dialog box. Use this parameter to direct the component to a specific portion of the test image. The component uses the value of the `testFlags` parameter to determine how it transforms large images before displaying them to the user.

testFlags

Constants (see below) that specify how the component is to display a test image that is larger than the test image portion of the dialog box. If you set this parameter to 0, the component uses a default method of its own choosing. In all cases, the component centers the area or point of interest in the test image portion of the dialog box, and then displays some part of the test image. See these constants:

`scPreferCropping`

`scPreferScaling`

`scPreferScalingAndCropping`

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

CompressMovies

ConvertToMovieJr

qtcompress

qtcompress.win

VideoProcessing

Declared In

`QuickTimeComponents.h`

TCFrameNumberToTimeCode

Converts a frame number into its corresponding timecode time value.

```
HandlerError TCFrameNumberToTimeCode (
   MediaHandler mh,
   long frameNumber,
   TimeCodeDef *tcdef,
   TimeCodeRecord *tcrec
);
```

Parameters

mh

The timecode media handler. You obtain this identifier by calling `GetMediaHandler`.

frameNumber

The frame number that is to be converted.

tcdef

A pointer to the `TimeCodeDef` structure to use for the conversion.

tcrec

A pointer to the `TimeCodeRecord` structure that is to receive the time value.

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
TimeCode Media Handlers

Declared In
`QuickTimeComponents.h`

TCGetCurrentTimeCode

Retrieves the timecode and source identification information for the current movie time.

```
HandlerError TCGetCurrentTimeCode (
   MediaHandler mh,
   long *frameNum,
   TimeCodeDef *tcdef,
   TimeCodeRecord *tcrec,
   UserData *srcRefH
);
```

Parameters

mh

The timecode media handler. You obtain this identifier by calling `GetMediaHandler`.

frameNum

A pointer to a field that is to receive the current frame number. Set this field to `NIL` if you don't want to retrieve the frame number.

tcdef

A pointer to a `TimeCodeDef` structure. The media handler returns the movie's timecode definition information. Set this parameter to `NIL` if you don't want this information.

tcrec

A pointer to a `TimeCodeRecord` structure. The media handler returns the current time value. Set this parameter to `NIL` if you don't want this information.

srcRefH

A pointer to a field that is to receive a handle containing the source information as a `UserDataRecord` structure. It is your responsibility to dispose of this structure when you are done with it. Set this field to `NIL` if you don't want this information.

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
QTKitTimeCode
qttimecode
qttimecode.win

Declared In
`QuickTimeComponents.h`

TCGetDisplayOptions

Retrieves the text characteristics that apply to timecode information displayed in a movie.

```
HandlerError TCGetDisplayOptions (
   MediaHandler mh,
   TCTextOptionsPtr textOptions
);
```

Parameters

mh

The timecode media handler. You obtain this identifier by calling `GetMediaHandler`.

textOptions

A pointer to a `TCTextOptions` structure. This structure will receive font and `style` information.

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
QTKitTimeCode

qttimecode
qttimecode.win
TimeCode Media Handlers

Declared In
`QuickTimeComponents.h`

TCGetSourceRef

Retrieves the source information from the timecode media sample reference.

```
HandlerError TCGetSourceRef (
   MediaHandler mh,
   TimeCodeDescriptionHandle tcdH,
   UserData *srefH
);
```

Parameters

mh

The timecode media handler. You obtain this identifier by calling `GetMediaHandler`.

tcdH

Specifies a handle to a `TimeCodeDescription` structure that defines the media sample reference for this operation.

srefH

Specifies a pointer to a handle that will receive the source information as a `UserDataRecord` structure. It is your application's responsibility to dispose of this structure when you are done with it.

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
`QuickTimeComponents.h`

TCGetTimeCodeAtTime

Returns a track's timecode information corresponding to a specific media time.

```
HandlerError TCGetTimeCodeAtTime (
   MediaHandler mh,
   TimeValue mediaTime,
   long *frameNum,
   TimeCodeDef *tcdef,
   TimeCodeRecord *tcdata,
   UserData *srcRefH
);
```

Parameters

mh

The timecode media handler. You obtain this identifier by calling `GetMediaHandler`.

mediaTime

A time value for which you want to retrieve timecode information. This time value is expressed in the media's time coordinate system.

frameNum

A pointer to a field that is to receive the current frame number. Set this field to `NIL` if you don't want to retrieve the frame number.

tcdef

A pointer to a `TimeCodeDef` structure. The media handler returns the movie's timecode definition information. Set this parameter to `NIL` if you don't want this information.

tcdata

A pointer to a `TimeCodeRecord` structure. The media handler returns the current time value. Set this parameter to `NIL` if you don't want this information.

srcRefH

A pointer to a field that is to receive a handle containing the source information as a `UserDataRecord` structure. It is your responsibility to dispose of this structure when you are done with it. Set this field to `NIL` if you don't want this information.

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
TimeCode Media Handlers

Declared In
`QuickTimeComponents.h`

TCGetTimeCodeFlags

Retrieves the timecode control flags.

```
HandlerError TCGetTimeCodeFlags (
   MediaHandler mh,
   long *flags
);
```

Parameters

mh

The timecode media handler. You obtain this identifier by calling `GetMediaHandler`.

flags

A pointer to a field that is to receive a control flag (see below). See these constants:

`tcdfShowTimeCode`

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
QTKitTimeCode
qttimecode
qttimecode.win

Declared In
`QuickTimeComponents.h`

TCSetDisplayOptions

Sets the text characteristics that apply to timecode information displayed in a movie.

```
HandlerError TCSetDisplayOptions (
   MediaHandler mh,
   TCTextOptionsPtr textOptions
);
```

Parameters

mh

The timecode media handler. You obtain this identifier by calling `GetMediaHandler`.

textOptions

A pointer to a `TCTextOptions` structure. This structure contains font and `style` information.

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
QTKitTimeCode
qttimecode
qttimecode.win

Declared In
`QuickTimeComponents.h`

TCSetSourceRef

Changes the source information in the timecode media sample reference.

```
HandlerError TCSetSourceRef (
   MediaHandler mh,
   TimeCodeDescriptionHandle tcdH,
   UserData srefH
);
```

Parameters

mh
The timecode media handler. You obtain this identifier by calling `GetMediaHandler`.

tcdH
Specifies a handle containing the timecode media sample reference that is to be updated.

srefH
Specifies a handle to the source information to be placed in the sample reference as a `UserDataRecord` structure. It is your application's responsibility to dispose of this structure when you are done with it.

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
QTKitTimeCode
qttimecode
qttimecode.win
TimeCode Media Handlers

Declared In
`QuickTimeComponents.h`

TCSetTimeCodeFlags

Changes the flag that affects how the toolbox handles timecode information.

```
HandlerError TCSetTimeCodeFlags (
   MediaHandler mh,
   long flags,
   long flagsMask
);
```

Parameters

mh

The timecode media handler. You obtain this identifier by calling `GetMediaHandler`.

flags

The new flag value. See these constants:

`tcdfShowTimeCode`

flagsMask

Specifies which of the flag values are to change. The media handler modifies only those flag values that correspond to bits that are set to 1 in this parameter. Use the flag values from the `flags` parameter. To turn off timecode display, set the `tcdfShowTimeCode` flag to 1 in the `flagsMask` parameter and to 0 in the `flags` parameter.

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

QTKitTimeCode

qttimecode

qttimecode.win

TimeCode Media Handlers

Declared In

`QuickTimeComponents.h`

TCTimeCodeToFrameNumber

Converts a timecode time value into its corresponding frame number.

```
HandlerError TCTimeCodeToFrameNumber (
   MediaHandler mh,
   TimeCodeDef *tcdef,
   TimeCodeRecord *tcrec,
   long *frameNumber
);
```

Parameters

mh

The timecode media handler. You obtain this identifier by calling `GetMediaHandler`.

tcdef

A pointer to the `TimeCodeDef` structure to use for the conversion.

tcrec

A pointer to the `TimeCodeRecord` structure containing the time value to convert.

frameNumber

A pointer to a field that is to receive the frame number that corresponds to the time value in the `tcrec` parameter.

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
QTKitTimeCode
qttimecode
qttimecode.win
TimeCode Media Handlers

Declared In
`QuickTimeComponents.h`

TCTimeCodeToString

Converts a time value into a text string (HH:MM:SS:FF).

```
HandlerError TCTimeCodeToString (
   MediaHandler mh,
   TimeCodeDef *tcdef,
   TimeCodeRecord *tcrec,
   StringPtr tcStr
);
```

Parameters

mh

The timecode media handler. You obtain this identifier by calling `GetMediaHandler`.

tcdef

A pointer to the `TimeCodeDef` structure to use for the conversion.

tcrec

A pointer to the `TimeCodeRecord` structure to use for the conversion.

tcStr

A pointer to a text string that is to receive the converted time value.

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

Discussion
If the timecode uses the dropframe technique, the separators are semicolons (;) rather than colons (:).

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
QTKitTimeCode
qttimecode
qttimecode.win
TimeCode Media Handlers

Declared In
`QuickTimeComponents.h`

TextExportGetDisplayData

Retrieves text display information for the current sample in the specified text export component.

```
ComponentResult TextExportGetDisplayData (
   TextExportComponent ci,
   TextDisplayData *textDisplay
);
```

Parameters

ci

Specifies the text export component for this operation. Applications can obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

textDisplay

Contains a pointer to a `TextDisplayData` structure. On return, this structure contains the display settings of the current text sample.

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

Discussion
You call this function to retrieve the text display data structure for a text sample. The text display data structure contains the formatting information for the text sample. When the text export component exports a text sample, it uses the information in this structure to generate the appropriate text descriptors for the sample. Likewise, when the text import component imports a text sample, it sets the appropriate fields in the text display data structure based on the sample's text descriptors.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`QuickTimeComponents.h`

TextExportGetSettings

Retrieves the value of the text export option for the specified text export component.

```
ComponentResult TextExportGetSettings (
   TextExportComponent ci,
   long *setting
);
```

Parameters

ci

Specifies the text export component for this operation. Applications can obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

setting

Contains a pointer to a 32-bit integer. On return, this integer contains a constant (see below) that represents the current value of the text export option. See these constants:

`kMovieExportTextOnly`

`kMovieExportAbsoluteTime`

`kMovieExportRelativeTime`

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
`QuickTimeComponents.h`

TextExportGetTimeFraction

Retrieves the time scale the specified text export component uses to calculate time stamps.

```
ComponentResult TextExportGetTimeFraction (
   TextExportComponent ci,
   long *movieTimeFraction
);
```

Parameters

ci

Specifies the text export component for this operation. Applications can obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

movieTimeFraction

Contains a pointer to a 32-bit integer. On return, this integer contains the time scale used in the fractional part of time stamps.

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

Discussion
You call this function to retrieve the time scale used by the text export component to calculate the fractional part of time stamps. You set a text component's time scale by specifying it in the Text Export Settings dialog box or by calling `TextExportSetTimeFraction` (page 112).

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`QuickTimeComponents.h`

TextExportSetSettings

Sets the value of the text export option for the specified text export component.

```
ComponentResult TextExportSetSettings (
   TextExportComponent ci,
   long setting
);
```

Parameters

ci

Specifies the text export component for this operation. Applications can obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

setting

A constant (see below) that specifies the new value of the text export option. See these constants:

`kMovieExportTextOnly`

`kMovieExportAbsoluteTime`

`kMovieExportRelativeTime`

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
`QuickTimeComponents.h`

TextExportSetTimeFraction

Sets the time scale the specified text export component uses to calculate time stamps.

```
ComponentResult TextExportSetTimeFraction (
   TextExportComponent ci,
   long movieTimeFraction
);
```

Parameters

ci

Specifies the text export component for this operation. Applications can obtain this reference from `OpenComponent` or `OpenDefaultComponent`.

movieTimeFraction

Specifies the time scale used in the fractional part of time stamps. The value should be between 1 and 10000, inclusive.

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

Discussion
You call this function to set the time scale used by the text export component to calculate the fractional part of time stamps. You can also set a text component's time scale by specifying it in the text export settings dialog box. You can retrieve a text component's time scale by calling `TextExportGetTimeFraction` (page 111).

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`QuickTimeComponents.h`

TweenerDoTween

Performs a tween operation.

```
ComponentResult TweenerDoTween (
   TweenerComponent tc,
   TweenRecord *tr
);
```

Parameters

tc

The tween component for this operation.

tr

A pointer to the `TweenRecord` structure for the tween operation.

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

Discussion
QuickTime calls this function to interpolate the data used during a tween operation. The `TweenRecord` structure contains complete information about the tween operation, including the start and end values for the operation and a percentage that indicates the progress towards completion of the tween sample. This function should use the information in the tween record to calculate the tweened value, and should call the data function specified in the tween record, passing it the tweened value.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`QuickTimeComponents.h`

TweenerInitialize

Initializes your tween component for a single tween operation.

```
ComponentResult TweenerInitialize (
   TweenerComponent tc,
   QTAtomContainer container,
   QTAtom tweenAtom,
   QTAtom dataAtom
);
```

Parameters

tc

The tween component for this operation.

container

The container that holds the atoms specified by the `tweenAtom` and `dataAtom` parameters.

tweenAtom

The atom that contains all parameters for defining this tween. This includes the data atom and any special atoms, such as an atom of type `'qdrg'`, that may be necessary.

dataAtom

The atom that contains the values to be tweened. This atom is a child of the atom specified by the `tweenAtom` parameter. This parameter is provided as a convenience; you can also call QT atom container functions to locate the data atom in the container.

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

Discussion
This function sets up the tween component when it is first used. In your implementation of this function, you can allocate storage and set up any structures that you need for the duration of a tween operation. Although the container that holds the data atom is available during each call to `TweenerDoTween` (page 113), you can improve the performance of your tween component by extracting the data to be used by the `TweenerDoTween` function in this function.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`QuickTimeComponents.h`

TweenerReset

Cleans up when the tween operation is finished.

```
ComponentResult TweenerReset (
   TweenerComponent tc
);
```

Parameters

tc

The tween component for this operation.

Return Value

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

Discussion

This function releases storage allocated by the tween component when the component is no longer being used. It should release any storage allocated by the `TweenerInitialize` (page 114) function and close or release any other resources used by the component. A tween component may receive a `TweenerInitialize` call after being reset.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`QuickTimeComponents.h`

Callbacks

MovieExportGetDataProc

Defines a data source for an export operation.

```
typedef OSErr (*MovieExportGetDataProcPtr) (void *refCon, MovieExportGetDataParams
 *params);
```

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

```
OSErr MyMovieExportGetDataProc (
    void                        *refCon,
    MovieExportGetDataParams    *params );
```

Parameters

refCon

Contains the value passed to `MovieExportAddDataSource` (page 30) in the `refCon` parameter

params

The sample request is made through a `MovieExportGetDataParams` structure.

Return Value

See `Error Codes`. Your callback should return `noErr` if there is no error.

Discussion
This callback is passed to `MovieExportAddDataSource` (page 30) to define a new data source for an export operation. The function is used by the exporting application to request source media data to be used in the export operation. For example, in a video export operation, frames of video data (either compressed or uncompressed) are provided. In a sound export operation, buffers of audio (either compressed or uncompressed) are provided.

Special Considerations

The data pointed to by `dataPtr` must remain valid until the next call to this function. The `MovieExportGetDataProc` callback is responsible for allocating and disposing of the memory associated with this data pointer.

Declared In
`QuickTimeComponents.h`

MovieExportGetPropertyProc

Returns parameters that determine the appropriate format for movie export data.

```
typedef OSErr (*MovieExportGetPropertyProcPtr) (void *refcon, long trackID, OSType
 propertyType, void *propertyValue);
```

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

```
OSErr MyMovieExportGetPropertyProc (
    void      *refcon,
    long      trackID,
    OSType    propertyType,
    void      *propertyValue );
```

Parameters

refcon
Contains the value passed to `MovieExportAddDataSource` (page 30) in the `refCon` parameter.

trackID
Specifies the value returned from `MovieExportAddDataSource` (page 30).

propertyType
Contains a pointer to the location of the requested property information.

propertyValue

Specifies the property being requested (see below). See these constants:

```
scSoundSampleRateType
scSoundSampleSizeType
scSoundChannelCountType
scSoundCompressionType
movieExportWidth
movieExportHeight
movieExportVideoFilter
scSpatialSettingsType
scTemporalSettingsType
scDataRateSettingsType
movieExportDuration
```

Return Value

See `Error Codes`. Your callback should return `noErr` if there is no error. If this function doesn't have a setting for a requested property, it should return an error.

Discussion

This function is passed to `MovieExportAddDataSource` (page 30) to define a new data source for an export operation. For example, a video export operation may call this function to determine the dimensions of the destination video track. The export component provides a default value for the property based on the source data format. For example, if no values for video track width and height properties were provided by the callback function, the dimensions of the source data would be used.

Declared In

`QuickTimeComponents.h`

SCModalFilterProc

Filter routine called when a user event occurs in a sequence compression modal dialog box.

```
typedef Boolean (*SCModalFilterProcPtr) (DialogPtr theDialog, EventRecord *theEvent,
 short *itemHit, long refcon);
```

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

```
Boolean MySCModalFilterProc (
    DialogPtr      theDialog,
    EventRecord    *theEvent,
    short          *itemHit,
    long           refcon );
```

Parameters

theDialog

A pointer to a dialog box.

theEvent

A pointer to an `EventRecord` structure that defines a user event.

itemHit

A pointer to an item ID number in the dialog box.

refcon

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.

Return Value
Return TRUE if the event was handled, FALSE otherwise.

Declared In
`QuickTimeComponents.h`

SCModalHookProc

Called whenever the user selects an item in the dialog box.

```
typedef short (*SCModalHookProcPtr) (DialogPtr theDialog, short itemHit, void
*params, long refcon);
```

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

```
short MySCModalHookProc (
    DialogPtr    theDialog,
    short        itemHit,
    void         *params,
    long         refcon );
```

Parameters

theDialog

A pointer to a dialog box.

itemHit

A pointer to an item ID number in the dialog box.

params

A pointer to your data area.

refcon

A reference constant that the client code supplies to your callback.

Return Value
Return TRUE if the event was handled, FALSE otherwise.

Discussion
You can use this callback to customize the operation of the standard image-compression dialog box. For example, you might want to support a custom button that activates a secondary dialog box. Another possibility would be to provide additional validation support when the user clicks OK.

Declared In
`QuickTimeComponents.h`

Data Types

GraphicImageMovieImportComponent

Represents a type used by the Movie Components API.

```
typedef ComponentInstance GraphicImageMovieImportComponent;
```

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

Declared In
`QuickTimeComponents.h`

HandlerError

Represents a type used by the Movie Components API.

```
typedef ComponentResult HandlerError;
```

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

Declared In
`Movies.h`

MovieExportComponent

Represents a type used by the Movie Components API.

```
typedef ComponentInstance MovieExportComponent;
```

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

Declared In
`QuickTimeComponents.h`

MovieExportGetDataUPP

Represents a type used by the Movie Components API.

```
typedef STACK_UPP_TYPE(MovieExportGetDataProcPtr) MovieExportGetDataUPP;
```

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

Declared In
`QuickTimeComponents.h`

MovieExportGetPropertyUPP

Represents a type used by the Movie Components API.

```
typedef STACK_UPP_TYPE(MovieExportGetPropertyProcPtr) MovieExportGetPropertyUPP;
```

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

Declared In
`QuickTimeComponents.h`

MovieImportComponent

Represents a type used by the Movie Components API.

```
typedef ComponentInstance MovieImportComponent;
```

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

Declared In
`QuickTimeComponents.h`

pnotComponent

Represents a type used by the Movie Components API.

```
typedef ComponentInstance pnotComponent;
```

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

Declared In
`QuickTimeComponents.h`

SCModalFilterUPP

Represents a type used by the Movie Components API.

```
typedef STACK_UPP_TYPE(SCModalFilterProcPtr) SCModalFilterUPP;
```

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

Declared In
`QuickTimeComponents.h`

SCModalHookUPP

Represents a type used by the Movie Components API.

```
typedef STACK_UPP_TYPE(SCModalHookProcPtr) SCModalHookUPP;
```

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

Declared In
`QuickTimeComponents.h`

SCParams

Provides data for the SCGetCompressionExtended function.

```
struct SCParams {
     long              flags;
     CodecType         theCodecType;
     CodecComponent    theCodec;
     CodecQ            spatialQuality;
     CodecQ            temporalQuality;
     short             depth;
     Fixed             frameRate;
     long              keyFrameRate;
     long              reserved1;
     long              reserved2;
 };
```

Fields
`flags`

Discussion
Flags (see below). See these constants:

- `scGetCompression`
- `scShowMotionSettings`
- `scSettingsChangedItem`

`theCodecType`

Discussion
A compressor type; see `Codec Identifiers`.

`theCodec`

Discussion
An instance of a compressor component, obtained by calling `OpenComponent` or `OpenDefaultComponent`.

`spatialQuality`

Discussion
Constants (see below) that determine image spatial quality. See these constants:

- `codecMinQuality`
- `codecLowQuality`
- `codecNormalQuality`
- `codecHighQuality`
- `codecMaxQuality`
- `codecLosslessQuality`

`temporalQuality`

Discussion
Constants (see below) that determine image temporal quality.

`depth`

Discussion
Image data depth.

`frameRate`

Discussion
The frame rate.

`keyFrameRate`

Discussion
The key frame rate.

`reserved1`

Discussion
Reserved.

`reserved2`

Discussion
Reserved.

Related Functions
`SCGetCompressionExtended` (page 89)

Declared In
`QuickTimeComponents.h`

TCTextOptions

Holds text font and style information.

```
struct TCTextOptions {
     short       txFont;
     short       txFace;
     short       txSize;
     short       pad;
     RGBColor    foreColor;
     RGBColor    backColor;
 };
```

Fields
`txFont`

Discussion
Specifies the number of the font.

`txFace`

Discussion
Specifies the font's `style` (bold, italic, and so on).

`txSize`

Discussion
Specifies the font's size.

`pad`

Discussion
Unused field to make structure long-word aligned.

`foreColor`

Discussion
Specifies the foreground color.

`backColor`

Discussion
Specifies the background color.

Related Functions
`TCGetDisplayOptions` (page 103)
`TCSetDisplayOptions` (page 106)

Declared In
`QuickTimeComponents.h`

TCTextOptionsPtr

Represents a type used by the Movie Components API.

```
typedef TCTextOptions * TCTextOptionsPtr;
```

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

Declared In
`QuickTimeComponents.h`

TextDisplayData

Contains formatting information for a text sample.

```
struct TextDisplayData {
     long         displayFlags;
     long         textJustification;
     RGBColor     bgColor;
     Rect         textBox;
     short        beginHilite;
     short        endHilite;
     RGBColor     hiliteColor;
     Boolean      doHiliteColor;
     SInt8        filler;
     TimeValue    scrollDelayDur;
     Point        dropShadowOffset;
     short        dropShadowTransparency;
 };
```

Fields

`displayFlags`

Discussion

Contains flags (see below) that represent the values of text descriptors. See these constants:

- `dfDontDisplay`
- `dfDontAutoScale`
- `dfClipToTextBox`
- `dfShrinkTextBoxToFit`
- `dfScrollIn`
- `dfScrollOut`
- `dfHorizScroll`
- `dfReverseScroll`

`textJustification`

Discussion

Contains constants (see below) that specify the alignment of the text in the text box. Possible values are `teFlushDefault`, `teCenter`, `teFlushRight`, and `teFlushLeft`. For more information on text alignment and the text justification constants, see the "TextEdit" chapter of *Inside Macintosh: Text*. See these constants:

`bgColor`

Discussion

Specifies the background color of the rectangle specified by the `textBox` field. The background color is specified as an RGB color value.

`textBox`

Discussion

Specifies the rectangle of the text box.

`beginHilite`

Discussion

Specifies the one-based index of the first character in the sample to highlight.

`endHilite`

Discussion

Specifies the one-based index of the last character in the sample to highlight.

`doHiliteColor`

Discussion
Specifies whether to use the color specified by the `hiliteColor` field for highlighting. If the `value` of this field is TRUE, the highlight color is used for highlighting. If the `value` of this field is FALSE, reverse video is used for highlighting.

`filler`

Discussion
Reserved.

`scrollDelayDur`

Discussion
Specifies a scroll delay. The scroll delay is specified as the number of units of delay in the text track's time scale. For example, if the time scale is 600, a scroll delay of 600 causes the sample text to be delayed one second. In order for this field to take effect, scrolling must be enabled.

`dropShadowOffset`

Discussion
Specifies an offset for the drop shadow. For example, if the point specified is (3,4), the drop shadow is offset 3 pixels to the right and 4 pixels down. In order for this field to take effect, drop shadowing must be enabled.

`dropShadowTransparency`

Discussion
Specifies the intensity of the drop shadow as a value between 0 and 255. In order for this field to take effect, drop shadowing must be enabled.

Discussion
When the text export component exports a text sample, it uses the information in this structure to generate the appropriate text descriptors for the sample. Likewise, when the text import component imports a text sample, it sets the appropriate fields in this structure based on the sample's text descriptors.

Related Functions
`TextExportGetDisplayData` (page 110)

Declared In
`QuickTimeComponents.h`

TextExportComponent

Represents a type used by the Movie Components API.

```
typedef ComponentInstance TextExportComponent;
```

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

Declared In
`QuickTimeComponents.h`

TimeCodeDef

Contains timecode format information.

```
struct TimeCodeDef {
     long         flags;
     TimeScale    fTimeScale;
     TimeValue    frameDuration;
     UInt8        numFrames;
     UInt8        padding;
 };
```

Fields

`flags`

Discussion

Contains flags (see below) that provide timecode format information. See these constants:

`tcDropFrame`

`tc24HourMax`

`tcNegTimesOK`

`tcCounter`

`fTimeScale`

Discussion

Contains the time scale for interpreting the `frameDuration` field. This field indicates the number of time units per second.

`frameDuration`

Discussion

Specifies how long each frame lasts, in the units defined by the `fTimeScale` field.

`numFrames`

Discussion

Indicates the number of frames stored per second. In the case of timecodes that are interpreted as counters, this field indicates the number of frames stored per timer "tick."

`padding`

Discussion

Unused.

Related Functions

`TCFrameNumberToTimeCode` (page 102)
`TCGetCurrentTimeCode` (page 102)
`TCGetTimeCodeAtTime` (page 104)
`TCTimeCodeToFrameNumber` (page 108)
`TCTimeCodeToString` (page 109)

Declared In

`QuickTimeComponents.h`

TimeCodeDescriptionHandle

Represents a type used by the Movie Components API.

```
typedef TimeCodeDescriptionPtr * TimeCodeDescriptionHandle;
```

Availability

Available in Mac OS X v10.0 and later.

Declared In
QuickTimeComponents.h

TimeCodeDescriptionPtr

Represents a type used by the Movie Components API.

```
typedef TimeCodeDescription * TimeCodeDescriptionPtr;
```

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

Declared In
QuickTimeComponents.h

TimeCodeRecord

Interprets time information as both a time value (HH:MM:SS:FF) and a frame count.

```
union TimeCodeRecord {
     TimeCodeTime       t;
     TimeCodeCounter    c;
 };
```

Fields
t

Discussion
The timecode value interpreted as time in a TimeCodeTime structure.

c

Discussion
The timecode value interpreted as a frame count in a TimeCodeCounter structure.

Discussion
When you use the timecode media handler to work with time values, the media handler uses TimeCodeRecord structures to store the time values. These structures allows you to interpret the time information as either a time value (HH:MM:SS:FF) or a counter value. Given a timecode definition, you can freely convert from frame numbers to time values and from time values to frame numbers. For a time value of 00:00:12:15 (HH:MM:SS:FF), you would obtain a frame number of 375 ((12*30) +15).

Related Functions
TCFrameNumberToTimeCode (page 102)
TCGetCurrentTimeCode (page 102)
TCGetTimeCodeAtTime (page 104)
TCTimeCodeToFrameNumber (page 108)
TCTimeCodeToString (page 109)

Declared In
QuickTimeComponents.h

TweenerComponent

Represents a type used by the Movie Components API.

```
typedef ComponentInstance TweenerComponent;
```

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

Declared In
`QuickTimeComponents.h`

TweenRecord

Passes information to your tween component's TweenDoTween method.

```
struct TweenRecord {
     long              version;
     QTAtomContainer   container;
     QTAtom            tweenAtom;
     QTAtom            dataAtom;
     Fixed             percent;
     TweenerDataUPP    dataProc;
     void *            private1;
     void *            private2;
 };
```

Fields
`version`

Discussion
The version number of this structure. This field is initialized to 0.

`container`

Discussion
The atom container that contains the tween data.

`tweenAtom`

Discussion
The atom for this tween entry's data in the container.

`percent`

Discussion
The percentage by which to change the data.

`dataProc`

Discussion
The procedure the tween component calls to send the tweened value to the receiving track.

`private1`

Discussion
Reserved.

`private2`

Discussion
Reserved.

Related Functions
`TweenerDataProc`
`TweenerDoTween` (page 113)

Declared In
`QuickTimeComponents.h`

Constants

MIDIImportSetSettings Values

Constants passed to MIDIImportSetSettings.

```
enum {
  kMIDIImportSilenceBefore      = 1 << 0,
  kMIDIImportSilenceAfter       = 1 << 1,
  kMIDIImport2OPlayable         = 1 << 2,
  kMIDIImportWantLyrics         = 1 << 3
};
```

Declared In
`QuickTimeComponents.h`

TextExportSetSettings Values

Constants passed to TextExportSetSettings.

```
enum {
  kMovieExportTextOnly          = 0,
  kMovieExportAbsoluteTime      = 1,
  kMovieExportRelativeTime      = 2
};
```

Declared In
`QuickTimeComponents.h`

movieExportDuration

Constants grouped with movieExportDuration.

```
enum {
  movieExportUseConfiguredSettings = 'ucfg', /* pointer to Boolean*/
  movieExportWidth              = 'wdth', /* pointer to Fixed*/
  movieExportHeight             = 'hegt', /* pointer to Fixed*/
  movieExportDuration           = 'dura', /* pointer to TimeRecord*/
  movieExportVideoFilter        = 'iflt', /* pointer to QTAtomContainer*/
  movieExportTimeScale          = 'tmsc' /* pointer to TimeScale*/
};
```

Constants

`movieExportWidth`

A fixed integer that represents a video track's image width in pixels.

Available in Mac OS X v10.0 and later.

Declared in `QuickTimeComponents.h`.

`movieExportHeight`

A fixed integer that represents a video track's image height in pixels.

Available in Mac OS X v10.0 and later.

Declared in `QuickTimeComponents.h`.

`movieExportDuration`

The `TimeRecord` structure for the whole movie.

Available in Mac OS X v10.0 and later.

Declared in `QuickTimeComponents.h`.

`movieExportVideoFilter`

A pointer to a `QTAtomContainer` handle that references a video track's filter atom container.

Available in Mac OS X v10.0 and later.

Declared in `QuickTimeComponents.h`.

Declared In

`QuickTimeComponents.h`

MovieImportDataRef Values

Constants passed to MovieImportDataRef.

```
enum {
  movieImportCreateTrack        = 1,
  movieImportInParallel         = 2,
  movieImportMustUseTrack       = 4,
  movieImportWithIdle           = 16
};
enum {
  movieImportResultUsedMultipleTracks = 8,
  movieImportResultNeedIdles    = 32,
  movieImportResultComplete     = 64
};
```

Constants

`movieImportResultNeedIdles`

Undocumented

Available in Mac OS X v10.0 and later.

Declared in `QuickTimeComponents.h`.

Declared In
`QuickTimeComponents.h`

Standard Compression Constants

Constants that represent constants for Standard Compression.

```
enum {
  /*
   * Indicates the client is ready to use the ICM compression session
   * API to perform compression operations. StdCompression disables
   * frame reordering and multi pass encoding if this flag is cleared.
   */
  scAllowEncodingWithCompressionSession = 1L << 8,
  /*
   * Indicates the client does not want the user to change the frame
   * reordering setting.
   */
  scDisableFrameReorderingItem  = 1L << 9,
  /*
   * Indicates the client does not want the user to change the multi
   * pass encoding setting
   */
  scDisableMultiPassEncodingItem = 1L << 10
};
enum {
  /*
   * Specifies if frame reordering can occur in encoding.
   */
  scVideoAllowFrameReorderingType = 'bfra', /* pointer to Boolean*/
  /*
   * The settings to control multi pass encoding.
   */
  scVideoMultiPassEncodingSettingsType = 'mpes' /* pointer to
SCVideoMutiPassEncodingSettings struct*/
};
enum {
  scListEveryCodec              = 1L << 1,
  scAllowZeroFrameRate          = 1L << 2,
  scAllowZeroKeyFrameRate       = 1L << 3,
  scShowBestDepth               = 1L << 4,
  scUseMovableModal             = 1L << 5,
  scDisableFrameRateItem        = 1L << 6,
  scShowDataRateAsKilobits      = 1L << 7
};
enum {
  scOKItem                      = 1,
  scCancelItem                  = 2,
  scCustomItem                  = 3
};
enum {
  scPositionRect                = 2,
  scPositionDialog              = 3,
  scSetTestImagePictHandle      = 4,
  scSetTestImagePictFile        = 5,
  scSetTestImagePixMap          = 6,
  scGetBestDeviceRect           = 7,
  scRequestImageSettings        = 10,
  scCompressImage               = 11,
  scCompressPicture             = 12,
  scCompressPictureFile         = 13,
  scRequestSequenceSettings     = 14,
  scCompressSequenceBegin       = 15,
  scCompressSequenceFrame       = 16,
  scCompressSequenceEnd         = 17,
```

```
  scDefaultPictHandleSettings   = 18,
  scDefaultPictFileSettings     = 19,
  scDefaultPixMapSettings       = 20,
  scGetInfo                     = 21,
  scSetInfo                     = 22,
  scNewGWorld                   = 23
};
enum {
  scPreferCropping              = 1 << 0,
  scPreferScaling               = 1 << 1,
  scPreferScalingAndCropping    = scPreferScaling | scPreferCropping,
  scDontDetermineSettingsFromTestImage = 1 << 2
};
enum {
  scSpatialSettingsType         = 'sptl', /* pointer to SCSpatialSettings struct*/
  scTemporalSettingsType        = 'tprl', /* pointer to SCTemporalSettings struct*/
  scDataRateSettingsType        = 'drat', /* pointer to SCDataRateSettings struct*/
  scColorTableType              = 'clut', /* pointer to CTabHandle*/
  scProgressProcType            = 'prog', /* pointer to ProgressRecord struct*/
  scExtendedProcsType           = 'xprc', /* pointer to SCExtendedProcs struct*/
  scPreferenceFlagsType         = 'pref', /* pointer to long*/
  scSettingsStateType           = 'ssta', /* pointer to Handle*/
  scSequenceIDType              = 'sequ', /* pointer to ImageSequence*/
  scWindowPositionType          = 'wndw', /* pointer to Point*/
  scCodecFlagsType              = 'cflg', /* pointer to CodecFlags*/
  scCodecSettingsType           = 'cdec', /* pointer to Handle*/
  scForceKeyValueType           = 'ksim', /* pointer to long*/
  scCompressionListType         = 'ctyl', /* pointer to OSType Handle*/
  scCodecManufacturerType       = 'cmfr', /* pointer to OSType*/
  scAvailableCompressionListType = 'avai', /* pointer to OSType Handle*/
  scWindowOptionsType           = 'shee', /* pointer to SCWindowSettings struct*/
  scSoundVBRCompressionOK       = 'cvbr', /* pointer to Boolean*/
  scSoundSampleRateChangeOK     = 'rcok', /* pointer to Boolean*/
  scSoundCompressionType        = 'ssct', /* pointer to OSType*/
  scSoundSampleRateType         = 'ssrt', /* pointer to UnsignedFixed*/
  scSoundInputSampleRateType    = 'ssir', /* pointer to UnsignedFixed*/
  scSoundSampleSizeType         = 'ssss', /* pointer to short*/
  scSoundChannelCountType       = 'sscc' /* pointer to short*/
};
enum {
  scTestImageWidth              = 80,
  scTestImageHeight             = 80
};
enum {
  scUserCancelled               = 1
};
enum {
  scWindowRefKindCarbon         = 'carb' /* WindowRef*/
};
```

Constants

`scVideoAllowFrameReorderingType`

Pointer to Boolean.

Available in Mac OS X v10.3 and later.

Declared in `QuickTimeComponents.h`.

`scSpatialSettingsType`
A video track's `SCSpatialSettings` structure.
Available in Mac OS X v10.0 and later.
Declared in `QuickTimeComponents.h`.

`scTemporalSettingsType`
A video track's `SCTemporalSettings` structure.
Available in Mac OS X v10.0 and later.
Declared in `QuickTimeComponents.h`.

`scDataRateSettingsType`
A video track's `SCDataRateSettings` structure.
Available in Mac OS X v10.0 and later.
Declared in `QuickTimeComponents.h`.

`scCodecSettingsType`
Pointer to `Handle`.
Available in Mac OS X v10.0 and later.
Declared in `QuickTimeComponents.h`.

`scForceKeyValueType`
Pointer to long.
Available in Mac OS X v10.0 and later.
Declared in `QuickTimeComponents.h`.

`scCodecManufacturerType`
Pointer to `OSType`.
Available in Mac OS X v10.0 and later.
Declared in `QuickTimeComponents.h`.

`scAvailableCompressionListType`
Pointer to `OSType Handle`.
Available in Mac OS X v10.2 and later.
Declared in `QuickTimeComponents.h`.

`scWindowOptionsType`
Pointer to `SCWindowSettings` struct.
Available in Mac OS X v10.3 and later.
Declared in `QuickTimeComponents.h`.

`scSoundVBRCompressionOK`
Pointer to Boolean.
Available in Mac OS X v10.2 and later.
Declared in `QuickTimeComponents.h`.

`scSoundSampleRateChangeOK`
Pointer to Boolean.
Available in Mac OS X v10.2 and later.
Declared in `QuickTimeComponents.h`.

`scSoundCompressionType`
A sound track's compression type constant; see `Codec Identifiers`.

Available in Mac OS X v10.0 and later.

Declared in `QuickTimeComponents.h`.

`scSoundSampleRateType`
An `UnsignedFixed` value that represents a sound track's sampling rate.

Available in Mac OS X v10.0 and later.

Declared in `QuickTimeComponents.h`.

`scSoundInputSampleRateType`
Pointer to `UnsignedFixed`.

Available in Mac OS X v10.2 and later.

Declared in `QuickTimeComponents.h`.

`scSoundSampleSizeType`
A short integer that represents a sound track's sample size.

Available in Mac OS X v10.0 and later.

Declared in `QuickTimeComponents.h`.

`scSoundChannelCountType`
A short integer that represents a sound track's channel count.

Available in Mac OS X v10.0 and later.

Declared in `QuickTimeComponents.h`.

Declared In
`QuickTimeComponents.h`

SCSetCompressFlags Values

Constants passed to SCSetCompressFlags.

```
enum {
  scCompressFlagIgnoreIdenticalFrames = 1
};
```

Declared In
`QuickTimeComponents.h`

SCParams Values

Constants passed to SCParams.

```
enum {
  scGetCompression              = 1,
  scShowMotionSettings          = 1L << 0,
  scSettingsChangedItem         = -1
};
```

Constants

`scGetCompression`

Undocumented

Available in Mac OS X v10.0 and later.

Declared in `QuickTimeComponents.h`.

`scShowMotionSettings`

Undocumented

Available in Mac OS X v10.0 and later.

Declared in `QuickTimeComponents.h`.

Declared In

`QuickTimeComponents.h`

TCSetTimeCodeFlags Values

Constants passed to TCSetTimeCodeFlags.

```
enum {
  tcdfShowTimeCode              = 1 << 0
};
```

Declared In

`QuickTimeComponents.h`

TimeCodeDef Values

Constants passed to TimeCodeDef.

```
enum {
  tcDropFrame                   = 1 << 0,
  tc24HourMax                   = 1 << 1,
  tcNegTimesOK                  = 1 << 2,
  tcCounter                     = 1 << 3
};
```

Constants

`tcDropFrame`

Indicates that the timecode drops frames occasionally to stay in synchronization. Some timecodes run at other than a whole number of frames per second. For example, NTSC video runs at 29.97 frames per second. In order to resynchronize between the timecode rate and a 30 frames-per-second playback rate, the timecode drops a frame at a predictable time (in much the same way that leap years keep the calendar synchronized).

Available in Mac OS X v10.0 and later.

Declared in `QuickTimeComponents.h`.

`tc24HourMax`

Indicates that the timecode values return to 0 at 24 hours.

Available in Mac OS X v10.0 and later.

Declared in `QuickTimeComponents.h`.

`tcNegTimesOK`

Indicates that the timecode supports negative time values.

Available in Mac OS X v10.0 and later.

Declared in `QuickTimeComponents.h`.

Declared In

`QuickTimeComponents.h`

Document Revision History

This table describes the changes to *Component Creation Reference for QuickTime*.

Date	Notes
2006-05-23	New document, based on previously published material, that describes the API for creating QuickTime movie import, export, and preview components.

Document Revision History

Index

C

D

G

H

M

N

P

S

T