QuickTime Movie Track and Media Reference

QuickTime > Media Types & Media Handlers

2006-11-10

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

Simultaneously published in the United States and Canada.

Contents

QuickTime Movie Track and Media Reference

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

Overview

Track and media management functions help with the construction and editing of QuickTime movies.

Functions by Task

Adding Samples to Media Structures

`AddMediaSample` (page 20)
Adds sample data and a description to a media.

`AddMediaSampleReference` (page 25)
Works with samples that have already been added to a movie data file.

`AddMediaSampleReferences` (page 27)
Adds groups of samples to a movie data file.

`BeginMediaEdits` (page 33)
Starts a media-editing session.

`EndMediaEdits` (page 51)
Ends a media-editing session.

`GetMediaPreferredChunkSize` (page 66)
Retrieves the maximum chunk size for a media.

`GetMediaSample` (page 67)
Returns a sample from a movie data file.

`GetMediaSampleReference` (page 73)
Obtains reference information about samples that are stored in a movie data file.

`GetMediaSampleReferences` (page 74)
Obtains reference information about groups of samples that are stored in a movie.

`SetMediaDefaultDataRefIndex` (page 127)
Specifies which of a media's data references is to be accessed during an editing session.

`SetMediaPreferredChunkSize` (page 131)
Specifies a maximum chunk size for a media.

Creating Tracks and Media Structures

`DisposeMovieTrack` (page 48)
Removes a track from a movie.

`DisposeTrackMedia` (page 50)
Removes a media from a track.

`NewMovieTrack` (page 112)
Creates a new movie track, without a media.

`NewTrackMedia` (page 114)
Creates a media for a new track.

Determining Movie Creation and Modification Time

`GetMediaCreationTime` (page 54)
Returns the creation date and time stored in a media.

`GetMediaModificationTime` (page 65)
Returns a media's modification date and time.

`GetTrackCreationTime` (page 88)
Returns a track's creation date and time.

`GetTrackModificationTime` (page 97)
Returns a track's modification date and time.

Disabling Movies and Tracks

`GetTrackEnabled` (page 93)
Determines whether a track is currently enabled.

`SetTrackEnabled` (page 135)
Enables or disables a track.

Editing Tracks

`AddEmptyTrackToMovie` (page 19)
Duplicates a track from a movie into the same movie or into another movie.

`CopyTrackSettings` (page 43)
Copies many settings from one track to another, overwriting the destination settings.

`DeleteTrackSegment` (page 47)
Removes a specified segment from a track.

`GetTrackEditRate` (page 92)
Returns the rate of the track edit of a specified track at an indicated time.

`InsertEmptyTrackSegment` (page 103)
Adds an empty segment to a track.

`InsertMediaIntoTrack` (page 104)
Inserts a reference to a media segment into a track.

`InsertTrackSegment` (page 107)
Copies data into a track.

`ScaleTrackSegment` (page 125)
Changes the duration of a segment of a track.

Enhancing Movie Playback Performance

`GetMediaShadowSync` (page 77)
Obsolete; no longer supported.

`GetTrackDisplayMatrix` (page 91)
Returns a matrix that is the concatenation of all matrices currently affecting the track's location, scaling, and so on, including the movie's matrix, the track's matrix, and the modifier matrix.

`SetMediaShadowSync` (page 133)
Obsolete; no longer supported.

Finding and Adding Samples

`AddMediaSample2` (page 23)
Adds sample data and a description to a media.

`ExtendMediaDecodeDurationToDisplayEndTime` (page 52)
Prepares a media for the addition of a completely new sequence of samples by ensuring that the media display end time is not later than the media decode end time.

`GetMediaAdvanceDecodeTime` (page 54)
Returns the advance decode time of a media.

`GetMediaDataSizeTime64` (page 57)
Determines the size, in bytes, of the sample data in a media segment.

`GetMediaDecodeDuration` (page 58)
Returns the decode duration of a media.

`GetMediaDisplayDuration` (page 59)
Returns the display duration of a media.

`GetMediaDisplayEndTime` (page 59)
Returns the display end time of a media.

`GetMediaDisplayStartTime` (page 60)
Returns the display start time of a media.

`MediaContainsDisplayOffsets` (page 109)
Tests whether a media contains display offsets.

`MediaDecodeTimeToSampleNum` (page 109)
Finds the sample for a specified decode time.

`MediaDisplayTimeToSampleNum` (page 110)
Finds the sample number for a specified display time.

`TrackTimeToMediaDisplayTime` (page 142)
Converts a track's time value to a display time value that is appropriate to the track's media, using the track's edit list.

High-Level Movie Editing Functions

`AddMovieSelection` (page 29)
Adds one or more tracks to a movie.

`ClearMovieSelection` (page 34)
Removes the segment of the movie that is defined by the current selection.

`CopyMovieSelection` (page 42)
Creates a new movie that contains the original movie's current selection.

`CutMovieSelection` (page 44)
Creates a new movie that contains the original movie's current selection.

`IsScrapMovie` (page 108)
Checks the system scrap to find out if it can translate any of the data into a movie.

`PasteHandleIntoMovie` (page 117)
Takes the contents of a specified handle, together with its type, and pastes it into a specified movie.

`PasteMovieSelection` (page 118)
Places the tracks from one movie into another movie.

`PutMovieIntoTypedHandle` (page 120)
Takes a movie, or a single track from within that movie, and converts it into a handle of a specified type.

Locating a Movie's Tracks and Media Structures

`GetMediaTrack` (page 79)
Determines the track that uses a specified media.

`GetMovieIndTrack` (page 82)
Determines the track identifier of a track, given the track's index value.

`GetMovieIndTrackType` (page 84)
Searches for all of a movie's tracks that share a given media type or media characteristic.

`GetMovieTrack` (page 85)
Determines the track identifier of a track, given the track's ID value.

`GetMovieTrackCount` (page 85)
Returns the number of tracks in a movie.

`GetTrackID` (page 94)
Determines a track's unique track ID value.

`GetTrackMedia` (page 96)
Determines the media that contains a track's sample data.

`GetTrackMovie` (page 97)
Determines the movie that contains a specified track.

Low-Level Movie Editing Functions

`CopyMovieSettings` (page 42)
Copies many settings from one movie to another, overwriting the destination settings in the process.

DeleteMovieSegment (page 45)
Removes a specified segment from a movie.

InsertEmptyMovieSegment (page 102)
Adds an empty segment to a movie.

InsertMovieSegment (page 106)
Copies part of one movie to another.

ScaleMovieSegment (page 124)
Changes the duration of a segment of a movie.

Manipulating Media Input Maps

GetMediaInputMap (page 63)
Returns a copy of the input map associated with a specified media.

SetMediaInputMap (page 129)
Replaces the media's existing input map with a given input map.

Movie Functions

ConvertFileToMovieFile (page 36)
Converts a file to a movie file and supports a user settings dialog box for import operations.

ConvertMovieToFile (page 39)
Takes a specified movie (or a single track within that movie) and converts it into a specified file and type, supporting a Save As dialog box.

Movie Posters and Movie Previews

GetTrackUsage (page 100)
Determines whether a track is used in a movie, its preview, its poster, or a combination of these.

SetTrackUsage (page 140)
Specifies whether a track is used in a movie, its preview, its poster, or a combination of these.

Movies and Your Event Loop

PtInMovie (page 118)
Determines whether a specified point lies in the region defined by a movie's final display boundary region after it has been clipped by the movie's display clipping region.

PtInTrack (page 119)
Determines whether a specified point lies in the region defined by a track's display boundary region after it has been clipped by the movie's final display clipping region.

Selecting Media Handlers

`GetDataHandler` (page 53)
Retrieves the best data handler component to use with a given data reference.

`GetMediaDataHandler` (page 55)
Determines a media's data handler.

`GetMediaDataHandlerDescription` (page 55)
Retrieves information about a media's data handler.

`GetMediaHandler` (page 61)
Obtains a reference to a media handler component.

`GetMediaHandlerDescription` (page 62)
Retrieves information about a media handler.

`SetMediaDataHandler` (page 127)
Assigns a data handler to a media.

`SetMediaHandler` (page 128)
Assigns a specific media handler to a track.

Undo for Movies

`DisposeMovieEditState` (page 48)
Disposes of an edit state.

`NewMovieEditState` (page 112)
Creates an edit state.

`UseMovieEditState` (page 143)
Returns a movie to the condition determined by an edit state created previously.

Undo for Tracks

`DisposeTrackEditState` (page 50)
Disposes of a movie's track edit state.

`NewTrackEditState` (page 114)
Creates a new edit state for a given track.

`UseTrackEditState` (page 144)
Returns a track to the condition determined by an edit state created previously.

Working With Alternate Tracks

`GetMediaLanguage` (page 65)
Returns a media's localized language or region code.

`GetMediaQuality` (page 66)
Returns a media's quality level value.

`GetTrackAlternate` (page 87)
Determines all the tracks in an alternate group.

SelectMovieAlternates (page 126)
Instructs the Movie Toolbox to select appropriate tracks immediately.

SetAutoTrackAlternatesEnabled (page 126)
Enables or disables automatic track selection by the Movie Toolbox.

SetMediaLanguage (page 130)
Sets a media's localized language or region code.

SetMediaQuality (page 131)
Sets a media's quality level value.

SetTrackAlternate (page 134)
Adds tracks to, or remove tracks from, alternate groups.

Working With Media Samples

GetMediaDataSize (page 56)
Determines the size, in bytes, of the sample data in a media segment.

GetMediaSampleCount (page 70)
Determines the number of samples in a media.

GetMediaSampleDescription (page 71)
Retrieves a SampleDescription structure from a media.

GetMediaSampleDescriptionCount (page 72)
Returns the number of sample descriptions in a media.

GetMovieDataSize (page 80)
Determines the size of the sample data in a segment of a movie.

GetTrackDataSize (page 88)
Determines the size, in bytes, of the sample data in a segment of a track.

MediaTimeToSampleNum (page 111)
Lets you find the sample that contains the data for a specified time.

SampleNumToMediaTime (page 123)
Finds the time at which a specified sample plays.

SetMediaSampleDescription (page 132)
Changes the contents of a particular SampleDescription structure of a specified media.

Working With Media Time

GetMediaDuration (page 60)
Returns the duration of a media.

GetMediaTimeScale (page 78)
Determines a media's time scale.

SetMediaTimeScale (page 133)
Sets a media's time scale.

Working With Movie Spatial Characteristics

`GetTrackDimensions` (page 90)
Determines a track's source rectangle.

`GetTrackLayer` (page 95)
Retrieves a track's layer.

`GetTrackMatrix` (page 95)
Retrieves a track's transformation matrix.

`SetTrackDimensions` (page 134)
Establishes a track's source rectangle.

`SetTrackLayer` (page 136)
Sets a track's layer.

`SetTrackMatrix` (page 137)
Establishes a track's transformation matrix.

Working With QuickTime Sample Tables

`AddSampleTableToMedia` (page 31)
Adds a sample table to a media.

`CopyMediaMutableSampleTable` (page 41)
Obtains information about sample references in a media in the form of a sample table.

Working With Sound Volume

`GetTrackVolume` (page 102)
Returns a track's current volume setting.

`SetTrackVolume` (page 141)
Sets a track's current volume.

Working With Track References

`AddTrackReference` (page 31)
Adds a new track reference to a track.

`DeleteTrackReference` (page 46)
Removes a track reference from a track.

`GetNextTrackReferenceType` (page 86)
Determines all of the track reference types that are defined for a given track.

`GetTrackReference` (page 98)
Retrieves the track identifier contained in an existing track reference.

`GetTrackReferenceCount` (page 99)
Determines how many track references of a given type exist for a track.

`SetTrackReference` (page 138)
Modifies an existing track reference.

Working With Track Sound

`GetTrackSoundLocalizationSettings` (page 100)
Returns a handle to a copy of the current 3D sound settings for a specified track.

`SetTrackSoundLocalizationSettings` (page 139)
Applies 3D sound effect data to a track.

Working With Track Time

`GetTrackDuration` (page 91)
Returns the duration of a track.

`GetTrackOffset` (page 98)
Determines the time difference between the start of a track and the start of the movie that contains the track.

`SetTrackOffset` (page 137)
Modifies the duration of the empty space that lies at the beginning of a track, thus changing the duration of the entire track.

`TrackTimeToMediaTime` (page 142)
Converts a track's time value to a time value that is appropriate to the track's media, using the track's edit list.

Working With User Data

`GetMediaUserData` (page 79)
Obtains access to a media's user data list.

`GetTrackUserData` (page 101)
Obtains access to a track's user data list.

Supporting Functions

`AddClonedTrackToMovie` (page 18)
Constructs a clone of an existing track in a movie.

`AddMediaSampleFromEncodedFrame` (page 25)
Adds sample data and description from an encoded frame to a media.

`AddMediaSampleReferences64` (page 28)
Provides a 64-bit version of AddMediaSampleReferences.

`ConvertDataRefToMovieDataRef` (page 35)
Converts a piece of data in a storage location to a movie file format and stores it in another storage location, supporting a user settings dialog box for import operations.

`ConvertMovieToDataRef` (page 37)
Converts a specified movie (or a single track within a movie) into a specified file format and stores it in a specified storage location.

`GetMediaDataSize64` (page 57)
Provides a 64-bit version of GetMediaDataSize.

GetMediaSample2 (page 69)
: Retrieves sample data from a media file.

GetMediaSampleReferences64 (page 76)
: Provides a 64-bit version of GetMediaSampleReferences.

GetMediaSyncSampleCount (page 78)
: Gets the number of sync samples in a media.

GetMovieDataSize64 (page 80)
: Provides a 64-bit version of GetMovieDataSize.

GetMovieImporterForDataRef (page 81)
: Gets the movie importer component for a movie.

GetTrackDataSize64 (page 89)
: Provides a 64-bit version of GetTrackDataSize.

GetTrackEditRate64 (page 93)
: Returns the rate of the track edit of a specified track at an indicated time.

OpenADataHandler (page 115)
: Opens a data handler component.

QTGetMIMETypeInfo (page 121)
: Retrieves information about a particular MIME type.

SampleNumToMediaDecodeTime (page 122)
: Finds the decode time for a specified sample.

SampleNumToMediaDisplayTime (page 122)
: Finds the display time for a specified sample.

Functions

AddClonedTrackToMovie

Constructs a clone of an existing track in a movie.

```
OSErr AddClonedTrackToMovie (
   Track srcTrack,
   Movie dstMovie,
   long flags,
   Track *dstTrack
);
```

Parameters

sourceTrack
: Indicates the track to be cloned. Your application obtains this track identifier from such functions as NewMovieTrack (page 112) and GetMovieTrack (page 85). This is the source of the sample table once the cloned track is constructed.

destinationMovie
: Indicates the movie where the cloned track should be created. Your application obtains this identifier from such functions as NewMovie, NewMovieFromFile, and NewMovieFromHandle. Currently, this must be the movie that contains the source track.

flags

Flags (see below) that determine how cloning should be performed. You currently must pass `kQTCloneShareSamples`. See these constants:

`kQTCloneShareSamples`

`kQTCloneDontCopyEdits`

dstTrack

The address of storage where a reference to the newly constructed track is returned. If the function fails, this storage is set to `NIL`.

Return Value

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

Special Considerations

Most QuickTime developers should never need to call this function.

Version Notes

Introduced in QuickTime 5.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Movies.h`

AddEmptyTrackToMovie

Duplicates a track from a movie into the same movie or into another movie.

```
OSErr AddEmptyTrackToMovie (
   Track srcTrack,
   Movie dstMovie,
   Handle dataRef,
   OSType dataRefType,
   Track *dstTrack
);
```

Parameters

srcTrack

The source track for this operation. Your application obtains this track identifier from such functions as `NewMovieTrack` (page 112) and `GetMovieTrack` (page 85).

dstMovie

The destination movie for this operation. This can be the same movie as the source track or a different movie.

dataRef

A handle to the `data` reference. The type of information stored in the handle depends upon the `data` reference type specified by `dataRefType`.

dataRefType

The type of data reference; see `Data References`. If the data reference is an alias, you must set the parameter to `rAliasType`, indicating that the reference is an alias.

dstTrack

The newly created track's identifier is returned in this parameter. If `AddEmptyTrackToMovie` fails, the resulting track identifier is set to `NIL`.

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

Discussion
This function returns a newly created, empty track. The newly created track has the same media type and track settings as the specified track. However, no data is copied from the source track to the new track. To copy data from the source track to the new track, use `InsertTrackSegment` (page 107) after calling `AddEmptyTrackToMovie`.

Version Notes
This function has been available since QuickTime 2.0.

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

Related Sample Code
bMoviePaletteCocoa
qtdataref
qtdataref.win
ThreadsImporter
ThreadsImportMovie

Declared In
`Movies.h`

AddMediaSample

Adds sample data and a description to a media.

```
OSErr AddMediaSample (
   Media theMedia,
   Handle dataIn,
   long inOffset,
   unsigned long size,
   TimeValue durationPerSample,
   SampleDescriptionHandle sampleDescriptionH,
   long numberOfSamples,
   short sampleFlags,
   TimeValue *sampleTime
);
```

Parameters

theMedia

The media for this operation. Your application obtains this media identifier from such functions as `NewTrackMedia` (page 114) and `GetTrackMedia` (page 96). See `Media Identifiers`.

dataIn

A handle to the sample data. The `AddMediaSample` function adds this data to the media specified by the parameter `theMedia`. You specify the number of bytes of sample data with the `size` parameter. You can use the `inOffset` parameter to specify a byte offset into the data referred to by this handle.

inOffset

Specifies an offset into the data referred to by the handle contained in the `dataIn` parameter. Set this parameter to 0 if there is no offset.

size

The number of bytes of sample data to be added to the `media`. This parameter indicates the total number of bytes in the sample data to be added to the media, not the number of bytes per sample. Use the `numberOfSamples` parameter to indicate the number of samples that are contained in the sample data.

durationPerSample

The duration of each sample to be added. You must specify this parameter in the media's time scale. For example, if you are adding sound that was sampled at 22 kHz to a media that contains a sound track with the same time scale, you would set `durationPerSample` to 1. Similarly, if you are adding video that was recorded at 10 frames per second to a video media that has a time scale of 600, you would set this parameter to 60 to add a single sample.

sampleDescriptionH

A handle to a `SampleDescription` structure. Some media structures may require sample descriptions. There are different descriptions for different types of samples. For example, a media that contains compressed video requires that you supply an `ImageDescription` structure. A media that contains sound requires that you supply a `SoundDescription` structure. If the media does not require a `SampleDescription` structure, set this parameter to `NIL`.

numberOfSamples

The number of samples contained in the sample data to be added to the media. The Movie Toolbox considers the `value` of this parameter as well as the value of the `size` parameter when it determines the size of each sample that it adds to the media. You should set the `value` of this parameter so that the resulting sample size represents a reasonable compromise between total data retrieval time and the overhead associated with input and output (I/O). You should also consider the speed of the data storage device; CD-ROM devices are much slower than hard disks, for example, and should therefore have a smaller sample size. For a video media, set a sample size that corresponds to the size of a frame. For a sound media, choose a number of samples that corresponds to between 0.5 and 1.0 seconds of sound. In general, you should not create groups of sound samples that are less than 2 KB in size or greater than 15 KB. Typically, a sample size of about 8 KB is reasonable for most storage devices.

sampleFlags

Contains flags (see below) that control the add operation. Set unused flags to 0. See these constants:

```
mediaSampleNotSync
```

sampleTime

A pointer to a time value. After adding the sample data to the media, the `AddMediaSample` function returns the time where the sample was inserted in the time value referred to by this parameter. If you don't want to receive this information, set this parameter to `NIL`.

Return Value

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

Discussion

Your application specifies the sample and the media for the operation. `AddMediaSample` updates the media so that it contains the sample data. One call to this function can add several samples to a media; however, all the samples must be the same size. Samples are always appended to the end of the media. Furthermore, the media duration is extended each time a sample is added.

```
// AddMediaSample coding example
```

```
// See "Discovering QuickTime," page 250
#define kSoundSampleDuration    1
#define kSyncSample             0
#define kTrackStart             0
#define kMediaStart             0
#define kFix1                   0x00010000
void CreateMySoundTrack (Movie movie)
{
    Track                       track;
    Media                       media;
    Handle                      hSound =NIL;
    SoundDescriptionHandle      hSoundDesc =NIL;
    long                        lDataOffset;
    long                        lDataSize;
    long                        lNumSamples;
    hSound =GetResource(soundListRsrc, 128);
    if (hSound ==NIL)
        return;
    hSoundDesc =(SoundDescriptionHandle)NewHandle(4);

    CreateMySoundDescription(hSound,
                            hSoundDesc,
                            &lDataOffset,
                            &lNumSamples,
                            &lDataSize);

    track =NewMovieTrack(movie, 0, 0, kFullVolume);
    media =NewTrackMedia(track, SoundMediaType,
                                FixRound((**hSoundDesc).sampleRate),
                                NIL, 0);
    BeginMediaEdits(media);
    AddMediaSample(media,
                    hSound,
                    lDataOffset,        // offset in data
                    lDataSize,
                    kSoundSampleDuration,   // duration of each sound
                                            //  sample
                    (SampleDescriptionHandle)hSoundDesc,
                    lNumSamples,
                    kSyncSample,        // self-contained samples
                    NIL);
    EndMediaEdits(media);
    InsertMediaIntoTrack(track,
                            kTrackStart,    // track start time
                            kMediaStart,    // media start time
                            GetMediaDuration(media),
                            kFix1);
    if (hSoundDesc !=NIL)
        DisposeHandle((Handle)hSoundDesc);
}
```

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
qteffects
qteffects.win
vrmakepano
VRMakePano Library
vrmakepano.win

Declared In
`Movies.h`

AddMediaSample2

Adds sample data and a description to a media.

```
OSErr AddMediaSample2 (
   Media theMedia,
   const UInt8 *dataIn,
   ByteCount size,
   TimeValue64 decodeDurationPerSample,
   TimeValue64 displayOffset,
   SampleDescriptionHandle sampleDescriptionH,
   ItemCount numberOfSamples,
   MediaSampleFlags sampleFlags,
   TimeValue64 *sampleDecodeTimeOut
);
```

Parameters

theMedia

The media for this operation. You obtain this media identifier from such functions as `NewTrackMedia` (page 114) and `GetTrackMedia` (page 96).

dataIn

A handle to the sample data. The function adds this data to the media specified by `theMedia`. You specify the number of bytes of sample data with the `size` parameter.

size

The number of bytes of sample data to be added to the `media`. This parameter indicates the total number of bytes in the sample data to be added to the media, not the number of bytes per sample. Use the `numberOfSamples` parameter to indicate the number of samples that are contained in the sample data.

decodeDurationPerSample

The duration of each sample to be added, representing the amount of time that passes while the sample data is being displayed. You must specify this parameter in the media's time scale. For example, if you are adding sound that was sampled at 22 kHz to a media that contains a sound track with the same time scale, you would set `durationPerSample` to 1. Similarly, if you are adding video that was recorded at 10 frames per second to a video media that has a time scale of 600, you would set this parameter to 60. Note that this is the duration per sample, regardless of the number of samples being added.

displayOffset

A 64-bit time value that specifies the offset between the decode time (the start time of the track plus the duration of all previous samples) and the display time. This value is normally zero unless the sample is frame reordering compressed video.

sampleDescriptionH

A handle to a `SampleDescription` structure. Some media structures may require sample descriptions. There are different descriptions for different types of samples. For example, a media that contains compressed video requires that you supply an `ImageDescription` structure. A media that contains sound requires that you supply a `SoundDescription` structure. If the media does not require a `SampleDescription` structure, set this parameter to `NIL`.

numberOfSamples

The number of samples contained in the sample data to be added to the media. The Movie Toolbox considers the `value` of this parameter as well as the value of the `size` parameter when it determines the size of each sample that it adds to the media. You should set the `value` of this parameter so that the resulting sample size represents a reasonable compromise between total data retrieval time and the overhead associated with input and output. You should also consider the speed of the data storage device; CD-ROM devices are much slower than hard disks, for example, and should therefore have a smaller sample size. For a video media, set a sample size that corresponds to the size of a frame. For a sound media, choose a number of samples that corresponds to between 0.5 and 1.0 seconds of sound. In general, you should not create groups of sound samples that are less than 2 KB in size or greater than 15 KB. Typically, a sample size of about 8 KB is reasonable for most storage devices.

sampleFlags

Flags that control the add operation; set unused flags to 0: `mediaSampleNotSync` Indicates that the sample to be added is not a sync sample. Set this flag to 1 if the sample is not a sync sample; set it to 0 if the sample is a sync sample. See these constants:

```
mediaSampleNotSync
```

sampleDecodeTimeOut

A pointer to a time value that represents the sample decode time. After adding the sample data to the media, the function returns in this parameter the time where the sample was inserted. If you don't want to receive this information, set this parameter to `NIL`.

Return Value

An error code. Returns `noErr` if there is no error. You can access Movie Toolbox error returns through `GetMoviesError` and `GetMoviesStickyError`, as well as in the function result.

Discussion

Your application specifies the sample and the media for the operation. This function updates the media so that it contains the sample data. One call to this function can add several samples to a media. This function replaces `AddMediaSample` (page 20); it adds 64-bit support and support for frame reordering video compression (display offset).

Availability

Available in Mac OS X v10.3 and later.

Related Sample Code

CaptureAndCompressIPBMovie

QTExtractAndConvertToMovieFile

QTKitTimeCode

SCAudioCompress

Declared In

`Movies.h`

AddMediaSampleFromEncodedFrame

Adds sample data and description from an encoded frame to a media.

```
OSErr AddMediaSampleFromEncodedFrame (
   Media theMedia,
   ICMEncodedFrameRef encodedFrame,
   TimeValue64 *sampleDecodeTimeOut
);
```

Parameters

theMedia

The media for this operation. You obtain this media identifier from such functions as `NewTrackMedia` (page 114) and `GetTrackMedia` (page 96)

encodedFrame

An encoded frame token returned by an `ICMCompressionSequence`.

sampleDecodeTimeOut

A pointer to a time value. After adding the sample data to the media, the function returns the decode time where the first sample was inserted in the time value referred to by this parameter. If you don't want to receive this information, set this parameter to NULL.

Return Value

An error code. Returns `noErr` if there is no error. You can access Movie Toolbox error returns through `GetMoviesError` and `GetMoviesStickyError`, as well as in the function result.

Discussion

This is a convenience API to make it easy to add frames emitted by new ICM compression functions to media. It can return these errors:

Availability

Available in Mac OS X v10.3 and later.

Related Sample Code

OpenGLCaptureToMovie

Quartz Composer QCTV

Declared In

`Movies.h`

AddMediaSampleReference

Works with samples that have already been added to a movie data file.

```
OSErr AddMediaSampleReference (
   Media theMedia,
   long dataOffset,
   unsigned long size,
   TimeValue durationPerSample,
   SampleDescriptionHandle sampleDescriptionH,
   long numberOfSamples,
   short sampleFlags,
   TimeValue *sampleTime
);
```

Parameters

theMedia

The media for this operation. Your application obtains this media identifier from such functions as `NewTrackMedia` (page 114) and `GetTrackMedia` (page 96). See `Media Identifiers`.

dataOffset

The offset into the movie data file. This parameter is used differently by each data handler. For example, for the standard HFS data handler, this parameter specifies the offset into the file. This parameter contains either data you add yourself or the data offset returned by `GetMediaSampleReference` (page 73).

size

The number of bytes of sample data to be identified by the reference. This parameter indicates the total number of bytes in the sample data, not the number of bytes per sample. Use `numberOfSamples` to indicate the number of samples that are contained in the reference.

durationPerSample

The duration of each sample in the reference. You must specify this parameter in the media's time scale. For example, if you are referring to sound that was sampled at 22 kHz in a media that contains a sound track with the same time scale, to add a reference to a single sample you would set `durationPerSample` to 1. Similarly, if you are referring to video that was recorded at 10 frames per second in a video media that has a time scale of 60, you would set this parameter to 6 to add a reference to a single sample.

sampleDescriptionH

A handle to a `SampleDescription` structure. Some media structures may require sample descriptions. There are different descriptions for different types of samples. For example, a media that contains compressed video requires that you supply an `ImageDescription` structure. A media that contains sound requires that you supply a sound description structure. If the media does not require a `SampleDescription` structure, set this parameter to `NIL`.

numberOfSamples

The number of samples contained in the reference. For details, see `AddMediaSample` (page 20). If the media does not require a `SampleDescription` structure, set this parameter to `NIL`.

sampleFlags

Contains flags (see below) that control the operation. Set unused flags to 0. See these constants:

```
mediaSampleNotSync
```

sampleTime

A pointer to a time value. After adding the reference to the media, the `AddMediaSampleReference` function returns the time where the reference was inserted in the time value referred to by this parameter. If you don't want to receive this information, set this parameter to `NIL`.

Return Value

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

Discussion
This function does not add sample data to the file or device that contains a media. Rather, it defines references to sample data that you previously added to a movie data file. Instead of actually writing out samples to disk, this function writes out references to existing samples, which you specify in `dataOffset` and the `size` parameter. As with `AddMediaSample` (page 20), your application specifies the media for the operation. Note that one reference may refer to more than one sample; all the samples described by a reference must be the same size. This function does not update the movie data file as part of the add operation. Therefore, your application does not have to call `BeginMediaEdits` (page 33) before calling `AddMediaSampleReference`.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
AlwaysPreview
qt3dtween
qt3dtween.win
SlideShowImporter
SlideShowImporter.win

Declared In
`Movies.h`

AddMediaSampleReferences

Adds groups of samples to a movie data file.

```
OSErr AddMediaSampleReferences (
   Media theMedia,
   SampleDescriptionHandle sampleDescriptionH,
   long numberOfSamples,
   SampleReferencePtr sampleRefs,
   TimeValue *sampleTime
);
```

Parameters

`theMedia`
The media for this operation. Your application obtains this media identifier from such functions as `NewTrackMedia` (page 114) and `GetTrackMedia` (page 96). See `Media Identifiers`.

`sampleDescriptionH`
A handle to a `SampleDescription` structure. Some media structures may require sample descriptions. There are different descriptions for different types of samples. For example, a media that contains compressed video requires that you supply an `ImageDescription` structure. A media that contains sound requires that you supply a sound description structure. If you don't want the `SampleDescription` structure, set this parameter to `NIL`.

`numberOfSamples`
The number of `SampleReferenceRecord` structures pointed to by the `sampleRefs` parameter. Each structure may contain one or more contiguous samples. For details, see `AddMediaSample` (page 20).

sampleRefs

A pointer to the number of `SampleReferenceRecord` structures specified by the `numberOfSamples` parameter.

sampleTime

A pointer to a time value. After adding the reference to the media, the `AddMediaSampleReferences` function returns the time where the reference was inserted, using the time scale referred to by this parameter. If you don't want to receive this information, set this parameter to `NIL`.

Return Value

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

Discussion

Using this function instead of `AddMediaSampleReference` (page 25) can greatly improve the performance of operations that involve adding a large number of samples to a movie at one time. `AddMediaSampleReferences` provides no capabilities that weren't previously available with `AddMediaSampleReference`.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Movies.h`

AddMediaSampleReferences64

Provides a 64-bit version of AddMediaSampleReferences.

```
OSErr AddMediaSampleReferences64 (
   Media theMedia,
   SampleDescriptionHandle sampleDescriptionH,
   long numberOfSamples,
   SampleReference64Ptr sampleRefs,
   TimeValue *sampleTime
);
```

Parameters

theMedia

The media for this operation. Your application obtains this media identifier from such functions as `NewTrackMedia` (page 114) and `GetTrackMedia` (page 96). See `Media Identifiers`.

sampleDescriptionH

A handle to a `SampleDescription` structure. Some media structures may require sample descriptions. There are different descriptions for different types of samples. For example, a media that contains compressed video requires that you supply an `ImageDescription` structure. A media that contains sound requires that you supply a sound description structure. If you don't want the `SampleDescription` structure, set this parameter to `NIL`.

numberOfSamples

The number of `SampleReference64Record` structures pointed to by the `sampleRefs` parameter. Each structure may contain one or more contiguous samples. For details, see `AddMediaSample` (page 20).

sampleRefs

A pointer to the number of `SampleReference64Record` structures specified by the `numberOfSamples` parameter.

sampleTime

A pointer to a time value. After adding the reference to the media, the `AddMediaSampleReferences` function returns the time where the reference was inserted, using the time scale referred to by this parameter. If you don't want to receive this information, set this parameter to `NIL`.

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

Discussion
The only difference between this function and `AddMediaSampleReferences` (page 27) is that the `sampleRefs` parameter points to SampleReference64Record structures instead of `SampleReferenceRecord` structures.

Special Considerations

New applications should use this function instead of the 32-bit version.

Version Notes
Introduced in QuickTime 4.

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

Related Sample Code
CreateMovieFromReferences

Declared In
`Movies.h`

AddMovieSelection

Adds one or more tracks to a movie.

```
void AddMovieSelection (
   Movie theMovie,
   Movie src
);
```

Parameters

theMovie

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

src

The source movie for this operation. `AddMovieSelection` adds the tracks from this movie to the destination movie. The function adds these tracks at the time specified by the current selection in the destination movie.

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

Discussion
This function scales the source movie so that it fits into the destination selection. If the current selection in the destination movie has a 0 duration, the Movie Toolbox adds the segment at the beginning of the current selection. The entire source movie is used regardless of the selection in the source movie. The Movie Toolbox removes any empty tracks from the destination movie after the add operation. If you have assigned a progress function to the destination movie, the Movie Toolbox calls that progress function during long add operations. Following is an example of using this function:

```
// AddMovieSelection coding example
// See "Discovering QuickTime," page 363
Movie           movie1;
TimeValue       lOldDuration;
Movie           movie2;
long            lIndex, lOrigTrackCount, lReferenceIndex;
Track           track, trackSprite;
// get the first track in original movie and position at the start
trackSprite =GetMovieIndTrack(movie1, 1);
SetMovieSelection(movie1, 0, 0);
// remove all tracks except video in modifier movie
for (lIndex =1; lIndex <=GetMovieTrackCount(movie2); lIndex++) {
    Track       track =GetMovieIndTrack(movie2, lIndex);
    OSType      dwType;
    GetMediaHandlerDescription(GetTrackMedia(track),
                                &dwType, NIL, NIL);
    if (dwType !=VideoMediaType) {
        DisposeMovieTrack(track);
        lIndex--;
    }
}
// add the modifier track to original movie
lOldDuration =GetMovieDuration(movie1);
AddMovieSelection(movie1, movie2);
DisposeMovie(movie2);
// truncate the movie to the length of the original track
DeleteMovieSegment(movie1, lOldDuration,
                   GetMovieDuration(movie1) - lOldDuration);
// associate the modifier track with the original sprite track
track =GetMovieIndTrack(movie1, lOrigTrackCount + 1);
AddTrackReference(trackSprite, track, kTrackModifierReference,
                   &lReferenceIndex);
```

Special Considerations

Some Movie Toolbox functions can take a long time to execute. For example, if you call `FlattenMovie` and specify a large movie, the Movie Toolbox must read and write all the sample data for the movie. During such operations you may wish to display some kind of progress indicator to the user. A progress function is an application-defined function that you can create to track the progress of time-consuming activities and keep the user informed.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
SlideShowImporter

SlideShowImporter.win

Declared In
`Movies.h`

AddSampleTableToMedia

Adds a sample table to a media.

```
OSErr AddSampleTableToMedia (
   Media theMedia,
   QTSampleTableRef sampleTable,
   SInt64 startSampleNum,
   SInt64 numberOfSamples,
   TimeValue64 *sampleDecodeTimeOut
);
```

Parameters

theMedia

The media for this operation. You obtain this media identifier from such functions as `NewTrackMedia` (page 114) and `GetTrackMedia` (page 96).

sampleTable

A reference to an opaque sample table object containing sample references to be added to the media.

startSampleNum

The sample number of the first sample reference in the sample table to be added to the media. The first sample's number is 1.

numberOfSamples

The number of sample references from the sample table to be added to the media.

sampleDecodeTimeOut

A pointer to a time value. After adding the sample references to the media, the function returns the decode time where the first sample was inserted in the time value referred to by this parameter. If you don't want to receive this information, set this parameter to NULL.

Return Value
An error code. Returns `noErr` if there is no error. You can access Movie Toolbox error returns through `GetMoviesError` and `GetMoviesStickyError`, as well as in the function result.

Discussion
This function can return these errors:

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

Declared In
`Movies.h`

AddTrackReference

Adds a new track reference to a track.

```
OSErr AddTrackReference (
   Track theTrack,
   Track refTrack,
   OSType refType,
   long *addedIndex
);
```

Parameters

theTrack

Identifies the track for this operation. Your application obtains this track identifier from such functions as `NewMovieTrack` (page 112) and `GetMovieTrack` (page 85).

refTrack

The track to be identified in the track reference.

refType

The type of reference.

addedIndex

A pointer to a long integer. The toolbox returns the index value assigned to the new track reference. If you don't want this information, set this parameter to `NIL`.

Return Value

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

Discussion

The following code snippet shows how `AddTrackReference` can be used to add a modifier track reference to a sprite track.

```
// AddTrackReference coding example
// See "Discovering QuickTime," page 363
Movie           movie1;
TimeValue       lOldDuration;
Movie           movie2;
long            lIndex, lOrigTrackCount, lReferenceIndex;
Track           track, trackSprite;
// get the first track in original movie and position at the start
trackSprite =GetMovieIndTrack(movie1, 1);
SetMovieSelection(movie1, 0, 0);
// remove all tracks except video in modifier movie
for (lIndex =1; lIndex <=GetMovieTrackCount(movie2); lIndex++) {
    Track       track =GetMovieIndTrack(movie2, lIndex);
    OSType      dwType;
    GetMediaHandlerDescription(GetTrackMedia(track),
                                &dwType, NIL, NIL);
    if (dwType !=VideoMediaType) {
        DisposeMovieTrack(track);
        lIndex--;
    }
}
// add the modifier track to original movie
lOldDuration =GetMovieDuration(movie1);
AddMovieSelection(movie1, movie2);
DisposeMovie(movie2);
// truncate the movie to the length of the original track
DeleteMovieSegment(movie1, lOldDuration,
                   GetMovieDuration(movie1) - lOldDuration);
// associate the modifier track with the original sprite track
```

```
track =GetMovieIndTrack(movie1, lOrigTrackCount + 1);
AddTrackReference(trackSprite, track, kTrackModifierReference,
                  &lReferenceIndex);
```

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
samplemakeeffectmovie
samplemakeeffectmovie.win
vrmakepano
VRMakePano Library
vrmakepano.win

Declared In
`Movies.h`

BeginMediaEdits

Starts a media-editing session.

```
OSErr BeginMediaEdits (
   Media theMedia
);
```

Parameters

`theMedia`

The media for this operation. Your application obtains this media identifier from such functions as `NewTrackMedia` (page 114) and `GetTrackMedia` (page 96). See `Media Identifiers`.

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

Discussion
Use `EndMediaEdits` (page 51) to end a media-editing session. You must call `BeginMediaEdits` before you add samples to a media with the `AddMediaSample` (page 20) function. You must also call `BeginMediaEdits` before calling `InsertTrackSegment` (page 107) if you wish `InsertTrackSegment` to copy media samples instead of copying the segment by reference.

```
// BeginMediaEdits coding example
// See "Discovering QuickTime," page 89
void CreateMyVideoTrack (Movie movie)
{
    Track   track;
    Media   media;
    Rect    rect ={0, 0, 100, 320};
    track =NewMovieTrack(movie,
                FixRatio(rect.right, 1),
                FixRatio(rect.bottom, 1),
                kNoVolume);
    media =NewTrackMedia(track,
```

```
                VideoMediaType,
                600,                            // video time scale
                NIL, NIL);
    BeginMediaEdits(media);
    MyAddVideoSamplesToMedia(media, &rect);     // assemble data
    EndMediaEdits(media);
    InsertMediaIntoTrack(track,
                0,                              // track start time
                0,                              // media start time
                GetMediaDuration(media),
                kFix1);                         // normal speed
}
```

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
qteffects
qteffects.win
vrmakepano
VRMakePano Library
vrmakepano.win

Declared In
`Movies.h`

ClearMovieSelection

Removes the segment of the movie that is defined by the current selection.

```
void ClearMovieSelection (
   Movie theMovie
);
```

Parameters

`theMovie`

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

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

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`Movies.h`

ConvertDataRefToMovieDataRef

Converts a piece of data in a storage location to a movie file format and stores it in another storage location, supporting a user settings dialog box for import operations.

```
OSErr ConvertDataRefToMovieDataRef (
   Handle inputDataRef,
   OSType inputDataRefType,
   Handle outputDataRef,
   OSType outputDataRefType,
   OSType creator,
   long flags,
   ComponentInstance userComp,
   MovieProgressUPP proc,
   long refCon
);
```

Parameters

inputDataRef

A data reference that specifies the storage location of the source data.

inputDataRefType

The type of the input data reference.

outputDataRef

A data reference that specified the storage location to receive the converted data.

outputDataRefType

The type of the output data reference.

creator

The creator type of the output storage location.

flags

Flags (see below) that control the operation of the dialog box. See these constants:

```
createMovieFileDeleteCurFile
movieToFileOnlyExport
movieFileSpecValid
showUserSettingsDialog
```

userComp

An instance of a component to be used for converting the movie data.

proc

A progress callback function; see `MovieProgressProc` in the QuickTime API Reference.

refCon

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

Return Value

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

Discussion

This function converts a piece of data in a storage location into a movie and stores into another storage location. Both the input and the output storage locations are specified through data references. If the storage location is on a local file system, the file will have the specified creator. If specified as such in the flags, the function displays a dialog box that lets the user to choose the output file and the export type. If an export component (or its instance) is specified in `userComp`, it will be used for the conversion operation.

Version Notes
Introduced in QuickTime 6.4.

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

Declared In
`Movies.h`

ConvertFileToMovieFile

Converts a file to a movie file and supports a user settings dialog box for import operations.

```
OSErr ConvertFileToMovieFile (
   const FSSpec *inputFile,
   const FSSpec *outputFile,
   OSType creator,
   ScriptCode scriptTag,
   short *resID,
   long flags,
   ComponentInstance userComp,
   MovieProgressUPP proc,
   long refCon
);
```

Parameters

`inputFile`

A pointer to the file system specification for the file to be converted into a movie file.

`outputFile`

A pointer to the file specification for the destination movie file.

`creator`

The creator value for the file if it is a new one.

`scriptTag`

The script in which the movie file should be converted. Use the Script Manager constant `smSystemScript` to use the system script; use the `smCurrentScript` constant to use the current script. See *Inside Macintosh: Text* for more information about scripts and script tags.

`resID`

A pointer to a field that is to receive the resource ID of the file to be converted. If you don't want to receive the resource ID, set this parameter to `NIL`.

`flags`

Contains flags (see below) that control movie file conversion and determine whether or not the user settings dialog box appears. See these constants:

`createMovieFileDeleteCurFile`

`movieToFileOnlyExport`

`movieFileSpecValid`

`showUserSettingsDialog`

userComp

Indicates a component or component instance of the movie export component you want to perform the conversion. Otherwise, set this parameter to 0 for the Movie Toolbox to choose the appropriate component. If you pass in a component instance, it will be used by `ConvertFileToMovieFile`. This allows you to communicate directly with the component before using this function to establish any conversion parameters. If you pass in a component ID, an instance is created and closed within this function.

proc

Points to your progress callback. To remove a movie's progress function, set this parameter to `NIL`. Set this parameter to -1 for the Movie Toolbox to provide a default progress function. See `MovieProgressProc` for the interface your progress callback must support.

refCon

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

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

Discussion
Use this function to specify an input file and convert it to a movie file. Because some conversions may take a nontrivial amount of time, you can pass a standard movie progress function in the `proc` and `refCon` parameters.

Special Considerations

Once you are finished working with a movie, you should release the resources used by the movie by calling `DisposeMovie`.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
ImportExportMovie
qtdataexchange
qtdataexchange.win

Declared In
`Movies.h`

ConvertMovieToDataRef

Converts a specified movie (or a single track within a movie) into a specified file format and stores it in a specified storage location.

```
OSErr ConvertMovieToDataRef (
   Movie m,
   Track onlyTrack,
   Handle dataRef,
   OSType dataRefType,
   OSType fileType,
   OSType creator,
   long flags,
   ComponentInstance userComp
);
```

Parameters

theMovie

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

onlyTrack

The track in the source movie, if you want to convert only a single track.

dataRef

A data reference that specifies the storage location to receive the converted movie data.

dataRefType

The type of data reference. This function currently supports only alias data references.

fileType

The Mac OS file type of the storage location, which determines the export format.

creator

The creator type of the storage location.

flags

Flags (see below) that control the operation of the dialog box. See these constants:

`showUserSettingsDialog`

`movieToFileOnlyExport`

`movieFileSpecValid`

userComp

An instance of the component to be used for converting the movie data.

Return Value

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

Discussion

If the storage location is on a local file system, the file will have the specified file type and the creator. If specified as such in the flags, the function displays a dialog box that lets the user choose the output file and the export type. If an export component (or its instance) is specified in the `userComp` parameter, it will be used to perform the conversion operation.

Version Notes

Introduced in QuickTime 6.4.

Availability

Available in Mac OS X v10.3 and later.

Related Sample Code

BackgroundExporter

Declared In
`Movies.h`

ConvertMovieToFile

Takes a specified movie (or a single track within that movie) and converts it into a specified file and type, supporting a Save As dialog box.

```
OSErr ConvertMovieToFile (
   Movie theMovie,
   Track onlyTrack,
   FSSpec *outputFile,
   OSType fileType,
   OSType creator,
   ScriptCode scriptTag,
   short *resID,
   long flags,
   ComponentInstance userComp
);
```

Parameters

theMovie

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

onlyTrack

The track within the source movie for this conversion operation. To specify all tracks, set the `value` of this parameter to 0.

outputFile

A pointer to the file specification for the destination file.

fileType

The data type of the destination file for the movie specified in the parameter `theMovie`.

creator

The creator value for the output file if it is a new one.

scriptTag

The script into which the movie should be converted if the output file is a new one. Use the Script Manager constant `smSystemScript` to use the system script; use the `smCurrentScript` constant to use the current script. See *Inside Macintosh: Text* for more information about scripts and script tags.

resID

A pointer to a field that is to receive the resource ID of the open movie. If you don't want to receive this information, set the `resID` parameter to `NIL`.

flags

Contains flags (see below) that control whether and how the Save As dialog box appears. See these constants:

`showUserSettingsDialog`
`movieToFileOnlyExport`
`movieFileSpecValid`

userComp

If you want a particular movie export component to perform the conversion, you may pass the component or an instance of that component in this parameter. Otherwise, set it to 0 to allow the Movie Toolbox to use the appropriate component. If you pass in a component instance, it is used by `ConvertMovieToFile`. This allows you to communicate directly with the component before making this call to establish any conversion parameters. If you pass in a component ID, an instance is created and closed within this call.

Return Value

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

Discussion

Your application controls whether a Save As dialog box appears by setting the value of the `flags` parameter. The dialog box lets the user specify the file name and type. Supported types include standard QuickTime movies, flattened movies, single-fork flattened movies, and any format that is supported by a movie data export component. The following code snippets show how to call `ConvertMovieToFile` to provide a simple export capability and how to save a sound-only QuickTime movie as a WAV file.

```
// Providing an export capability with ConvertMovieToFile
err =ConvertMovieToFile (theMovie,     /* identifies movie */
                    NIL,                /* all tracks */
                    NIL,                /* no output file */
                    0,                  /* no file type */
                    0,                  /* no creator */
                    -1,                 /* script */
                    NIL,                /* no resource ID */
                    createMovieFileDeleteCurFile |
                        showUserSettingsDialog |
                        movieToFileOnlyExport,
                    0);                 /* no specific component */
// Saving a sound-only QuickTime movie as a WAVE file
// See "Discovering QuickTime," page 257
void SndSnip_SaveSoundMovieAsWAVEFile (Movie theMovie)
{
    StandardFileReply   myReply;
    // have the user select the name and location of the new WAVE file
    StandardPutFile("\pSave sound movie file as:",
                                        "\pUntitled.wav", &myReply);
    if (!myReply.sfGood)
        return;
    // use the default progress procedure, if any
    SetMovieProgressProc(theMovie, (MovieProgressUPP)-1L, 0);
    // export the movie into a file
    ConvertMovieToFile( theMovie,               // the movie to convert
                        NIL,                   // all tracks in the movie
                        &myReply.sfFile,        // the output file
                        kQTFileTypeWave,        // the output file type
                        FOUR_CHAR_CODE('TVOD'), // the output file creator
                        smSystemScript,         // the script
                        NIL,                   // no resource ID
                                                //   to be returned
                        0L,                     // no flags
                        NIL);                  // no specific component
}
```

Version Notes

Introduced in QuickTime 3 or earlier.

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

Related Sample Code
ImportExportMovie
MovieToAIFF
qtdataexchange
soundsnippets
soundsnippets.win

Declared In
`Movies.h`

CopyMediaMutableSampleTable

Obtains information about sample references in a media in the form of a sample table.

```
OSErr CopyMediaMutableSampleTable (
   Media theMedia,
   TimeValue64 startDecodeTime,
   TimeValue64 *sampleStartDecodeTime,
   SInt64 maxNumberOfSamples,
   TimeValue64 maxDecodeDuration,
   QTMutableSampleTableRef *sampleTableOut
);
```

Parameters

theMedia
The media for this operation. You obtain this media identifier from such functions as `NewTrackMedia` (page 114) and `GetTrackMedia` (page 96).

startDecodeTime
A 64-bit time value that represents the starting decode time of the sample references to be retrieved. You must specify this value in the media's time scale.

sampleStartDecodeTime
A pointer to a time value. The function updates this time value to indicate the actual decode time of the first returned sample reference. If you are not interested in this information, set this parameter to NULL. The returned time may differ from the time you specified with the `startDecodeTime` parameter. This will occur if the time you specified falls in the middle of a sample.

maxNumberOfSamples
A 64-bit signed integer that contains the maximum number of sample references to be returned. If you set this parameter to 0, the Movie Toolbox uses a value that is appropriate to the media.

maxDecodeDuration
A 64-bit time value that represents the maximum decode duration to be returned. The function does not return samples with greater decode duration than you specify with this parameter. If you set this parameter to 0, the Movie Toolbox uses a value that is appropriate for the media.

sampleTableOut
A reference to an opaque sample table object. When you are done with the returned sample table, release it with `QTSampleTableRelease`.

Return Value
An error code. Returns `memFullErr` if it could not allocate memory, `paramErr` if there was an invalid parameter, or `noErr` if there is no error. You can access Movie Toolbox error returns through `GetMoviesError` and `GetMoviesStickyError`, as well as in the function result.

Discussion
To find out how many samples were returned in the sample table, call `QTSampleTableGetNumberOfSamples`.

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

Related Sample Code
MovieVideoChart

Declared In
`Movies.h`

CopyMovieSelection

Creates a new movie that contains the original movie's current selection.

```
Movie CopyMovieSelection (
   Movie theMovie
);
```

Parameters

`theMovie`

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

Return Value
The new movie.

Discussion
This function creates a new movie from the source movie's current selection, but does not change the source movie or the selection. If you have assigned a progress function to the source movie, the Movie Toolbox calls that progress function during long copy operations.

Special Considerations

Your application must dispose of the new movie once you are done with it, using `DisposeMovie`.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`Movies.h`

CopyMovieSettings

Copies many settings from one movie to another, overwriting the destination settings in the process.

```
OSErr CopyMovieSettings (
   Movie srcMovie,
   Movie dstMovie
);
```

Parameters

srcMovie

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

dstMovie

The destination movie for this operation. The `CopyMovieSettings` function uses the settings from the source movie, which is specified by the `srcMovie` parameter, to replace the current settings of this movie.

Return Value

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

Discussion

Use this function to copy certain important settings from one movie to another. It copies the preferred rate and volume, source clipping region, matrix information, and user data; it does not copy the movie's contents. To work with movie contents, you should use segment editing functions.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

ConvertToMovieJr

MakeEffectMovie

qteffects.win

samplemakeeffectmovie

samplemakeeffectmovie.win

Declared In

`Movies.h`

CopyTrackSettings

Copies many settings from one track to another, overwriting the destination settings.

```
OSErr CopyTrackSettings (
   Track srcTrack,
   Track dstTrack
);
```

Parameters

srcTrack

The source track for this operation. Your application obtains this track identifier from such functions as `NewMovieTrack` (page 112) and `GetMovieTrack` (page 85).

dstTrack

The destination track for this operation. The `CopyTrackSettings` function uses the settings from the source track, which you specify with the `srcTrack` parameter, to replace the current settings of this track.

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

Discussion
This function copies matrix information, track volume, the clipping region, user data, matte information, media language, quality, user data, and other media-specific settings (such as sound balance and video graphics mode). It does not copy any alternate group information pertaining to the track. This function does not copy the track's contents. To work with track contents, you should use segment-editing functions.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
BurntTextSampleCode
qtaddeffectseg.win
qteffects
qteffects.win
qtstreamsplicer.win

Declared In
`Movies.h`

CutMovieSelection

Creates a new movie that contains the original movie's current selection.

```
Movie CutMovieSelection (
   Movie theMovie
);
```

Parameters

theMovie

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

Return Value
The newly created movie.

Discussion
This function removes the current selection from the original movie and makes the selection into a new movie. After the current selection has been removed from the original movie, the duration of the current selection is 0. The starting time of the current selection is not affected. If you have assigned a progress function to the source movie, the Movie Toolbox calls that progress function during long cut operations.

Special Considerations

Your application must dispose of the new movie once you are done with it, using `DisposeMovie`.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Movies.h`

DeleteMovieSegment

Removes a specified segment from a movie.

```
OSErr DeleteMovieSegment (
   Movie theMovie,
   TimeValue startTime,
   TimeValue duration
);
```

Parameters

theMovie

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

startTime

A time value specifying the starting point of the segment to be deleted.

duration

A time value that specifies the duration of the segment to be deleted.

Return Value

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

Discussion

You identify the segment to remove by specifying its starting time and duration. The following code snippet shows `DeleteMovieSegment` being used while adding a modifier track to a movie.

```
// DeleteMovieSegment coding example
// See "Discovering QuickTime," page 363
Movie           movie1;
TimeValue       lOldDuration;
Movie           movie2;
long            lIndex, lOrigTrackCount, lReferenceIndex;
Track           track, trackSprite;
// get the first track in original movie and position at the start
trackSprite =GetMovieIndTrack(movie1, 1);
SetMovieSelection(movie1, 0, 0);
// remove all tracks except video in modifier movie
for (lIndex =1; lIndex <=GetMovieTrackCount(movie2); lIndex++) {
    Track       track =GetMovieIndTrack(movie2, lIndex);
    OSType      dwType;
    GetMediaHandlerDescription(GetTrackMedia(track),
                               &dwType, NIL, NIL);
```

```
    if (dwType !=VideoMediaType) {
        DisposeMovieTrack(track);
        lIndex--;
    }
}
// add the modifier track to original movie
lOldDuration =GetMovieDuration(movie1);
AddMovieSelection(movie1, movie2);
DisposeMovie(movie2);
// truncate the movie to the length of the original track
DeleteMovieSegment(movie1, lOldDuration,
                   GetMovieDuration(movie1) - lOldDuration);
// associate the modifier track with the original sprite track
track =GetMovieIndTrack(movie1, lOrigTrackCount + 1);
AddTrackReference(trackSprite, track, kTrackModifierReference,
                   &lReferenceIndex);
```

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
qtspritesplus.win

Declared In
`Movies.h`

DeleteTrackReference

Removes a track reference from a track.

```
OSErr DeleteTrackReference (
   Track theTrack,
   OSType refType,
   long index
);
```

Parameters

theTrack
: Identifies the track for this operation. Your application obtains this track identifier from such functions as `NewMovieTrack` (page 112) and `GetMovieTrack` (page 85).

refType
: The type of reference.

index
: The index value of the reference to be deleted. You obtain this index value when you create the track reference.

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

Discussion
This function deletes a track reference from a track. If there are additional track references with higher index values, the toolbox automatically renumbers those references, decrementing their index values by 1.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
MakeEffectMovie
QTKitTimeCode
qttext
qttext.win
vrmovies.win

Declared In
`Movies.h`

DeleteTrackSegment

Removes a specified segment from a track.

```
OSErr DeleteTrackSegment (
   Track theTrack,
   TimeValue startTime,
   TimeValue duration
);
```

Parameters

theTrack
The track for this operation. Your application obtains this track identifier from such functions as `NewMovieTrack` (page 112) and `GetMovieTrack` (page 85).

startTime
A time value specifying the starting point of the segment to be deleted. This time value must be expressed in the time scale of the movie that contains the source track.

duration
A time value that specifies the duration of the segment to be deleted. This time value must be expressed in the time scale of the movie that contains the source track.

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

Discussion
You identify the segment to remove by specifying its starting time and duration.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
addhtactions.win
CaptureAndCompressIPBMovie
qttext
qttext.win
qtwiredactions

Declared In
`Movies.h`

DisposeMovieEditState

Disposes of an edit state.

```
OSErr DisposeMovieEditState (
   MovieEditState state
);
```

Parameters

state

The edit state for this operation. Your application obtains this edit state identifier when you create the edit state by calling `NewMovieEditState` (page 112).

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

Special Considerations

You must dispose of a movie's edit states before you dispose of the movie itself.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`Movies.h`

DisposeMovieTrack

Removes a track from a movie.

```
void DisposeMovieTrack (
   Track theTrack
);
```

Parameters

theTrack

The track for this operation. Your application obtains this track identifier from such functions as `NewMovieTrack` (page 112) and `GetMovieTrack` (page 85).

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

Discussion
The following code snippet illustrates the use of `DisposeMovieTrack`:

```
// DisposeMovieTrack coding example
// See "Discovering QuickTime," page 363
Movie           movie1;
TimeValue       lOldDuration;
Movie           movie2;
long            lIndex, lOrigTrackCount, lReferenceIndex;
Track           track, trackSprite;
// get the first track in original movie and position at the start
trackSprite =GetMovieIndTrack(movie1, 1);
SetMovieSelection(movie1, 0, 0);
// remove all tracks except video in modifier movie
for (lIndex =1; lIndex <=GetMovieTrackCount(movie2); lIndex++) {
    Track       track =GetMovieIndTrack(movie2, lIndex);
    OSType      dwType;
    GetMediaHandlerDescription(GetTrackMedia(track),
                               &dwType, NIL, NIL);
    if (dwType !=VideoMediaType) {
        DisposeMovieTrack(track);
        lIndex--;
    }
}
// add the modifier track to original movie
lOldDuration =GetMovieDuration(movie1);
AddMovieSelection(movie1, movie2);
DisposeMovie(movie2);
// truncate the movie to the length of the original track
DeleteMovieSegment(movie1, lOldDuration,
                   GetMovieDuration(movie1) - lOldDuration);
// associate the modifier track with the original sprite track
track =GetMovieIndTrack(movie1, lOrigTrackCount + 1);
AddTrackReference(trackSprite, track, kTrackModifierReference,
                   &lReferenceIndex);
```

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
AlwaysPreview
QTKitTimeCode
qttext
qttext.win
qttimecode.win

Declared In
`Movies.h`

DisposeTrackEditState

Disposes of a movie's track edit state.

```
OSErr DisposeTrackEditState (
   TrackEditState state
);
```

Parameters

state

The edit state for this operation. Your application obtains this edit state identifier when you create the edit state by calling the `NewTrackEditState` (page 114) function.

Return Value

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

Discussion

Your application must dispose of any edit states you create. You create an edit state by calling `NewTrackEditState` (page 114).

Special Considerations

You must dispose of a movie's track edit states before you dispose of the track or the movie.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Movies.h`

DisposeTrackMedia

Removes a media from a track.

```
void DisposeTrackMedia (
   Media theMedia
);
```

Parameters

theMedia

The media for this operation. Your application obtains this media identifier from such functions as `NewTrackMedia` (page 114) and `GetTrackMedia` (page 96). See `Media Identifiers`.

Return Value

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

Discussion

This function does not remove the track from its movie.

Version Notes

Introduced in QuickTime 3 or earlier.

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

Related Sample Code
QTKitTimeCode

Declared In
`Movies.h`

EndMediaEdits

Ends a media-editing session.

```
OSErr EndMediaEdits (
   Media theMedia
);
```

Parameters

`theMedia`

The media for this operation. Your application obtains this media identifier from such functions as `NewTrackMedia` (page 114) and `GetTrackMedia` (page 96). See `Media Identifiers`.

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

Discussion
The following code sample illustrates the use of `EndMediaEdits`:

```
// EndMediaEdits coding example
// See "Discovering QuickTime," page 89
void CreateMyVideoTrack (Movie movie)
{
    Track   track;
    Media   media;
    Rect    rect ={0, 0, 100, 320};
    track =NewMovieTrack(movie,
                FixRatio(rect.right, 1),
                FixRatio(rect.bottom, 1),
                kNoVolume);
    media =NewTrackMedia(track,
                VideoMediaType,
                600,                            // video time scale
                NIL, NIL);
    BeginMediaEdits(media);
    MyAddVideoSamplesToMedia(media, &rect);     // assemble data
    EndMediaEdits(media);
    InsertMediaIntoTrack(track,
                0,                              // track start time
                0,                              // media start time
                GetMediaDuration(media),
                kFix1);                         // normal speed
}
```

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
qteffects
qteffects.win
vrmakepano
VRMakePano Library
vrmakepano.win

Declared In
`Movies.h`

ExtendMediaDecodeDurationToDisplayEndTime

Prepares a media for the addition of a completely new sequence of samples by ensuring that the media display end time is not later than the media decode end time.

```
OSErr ExtendMediaDecodeDurationToDisplayEndTime (
   Media theMedia,
   Boolean *mediaChanged
);
```

Parameters

theMedia

The media for this operation. You obtain this media identifier from such functions as `NewTrackMedia` (page 114) and `GetTrackMedia` (page 96).

mediaChanged

A pointer to a Boolean that returns TRUE if any samples in the media were adjusted, FALSE otherwise. If you don't want to receive this information, set this parameter to NULL.

Return Value
An error code. Returns `memFullErr` if it could not allocate memory, `paramErr` if there was an invalid parameter, or `noErr` if there is no error. You can access Movie Toolbox error returns through `GetMoviesError` and `GetMoviesStickyError`, as well as in the function result.

Discussion
After adding a complete, well-formed set of samples to a media, the media's display end time should be the same as the media's decode end time (also called the media decode duration). However, this is not necessarily the case after individual sample-adding operations, and hence it is possible for a media to be left with a display end time later than its decode end time (if adding a sequence of frames is aborted halfway, for example).

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

Related Sample Code
CaptureAndCompressIPBMovie
OpenGLCaptureToMovie
Quartz Composer QCTV

Declared In
`Movies.h`

GetDataHandler

Retrieves the best data handler component to use with a given data reference.

```
Component GetDataHandler (
   Handle dataRef,
   OSType dataHandlerSubType,
   long flags
);
```

Parameters

dataRef

A handle to the `data` reference. The type of information stored in the handle depends upon the `data` reference type specified by the `dataHandlerSubType` parameter.

dataHandlerSubType

Identifies both the type of data reference and, by implication, the `component` subtype value assigned to the data handler components that operate on data references of that type.

flags

Contains flags (see below) that indicate the way in which you intend to use the data handler component. Note that not all data handlers necessarily support all services; for example, some data handler components may not support streaming writes. Set the appropriate flags to 1. See these constants:

`kDataHCanRead`

`kDataHCanWrite`

`kDataHCanStreamingWrite`

Return Value

The best data handler component conforming to the parameters passed in.

Discussion

Once you have used this function to get information about the best data handler component for your data reference, you can open and use the component using Component Manager functions. If the function returns a value of `NIL`, the toolbox was unable to find an appropriate data handler component. For more information about the error that caused a return of `NIL`, call `GetMoviesError`.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

qtdataref

qtfiletransfer

qtfiletransfer.win

ThreadsImporter

ThreadsImportMovie

Declared In

`Movies.h`

GetMediaAdvanceDecodeTime

Returns the advance decode time of a media.

```
TimeValue64 GetMediaAdvanceDecodeTime (
   Media theMedia
);
```

Parameters

theMedia

The media for this operation. You obtain this media identifier from such functions as `NewTrackMedia` (page 114) and `GetTrackMedia` (page 96).

Return Value

A 64-bit time value that represents the media's advance decode time. A media's advance decode time is the absolute value of the greatest-magnitude negative display offset of its samples, or 0 if there are no samples with negative display offsets. This is the amount that the decode time axis must be adjusted ahead of the display time axis to ensure that no sample's adjusted decode time is later than its display time. For media without nonzero display offsets, the advance decode time is 0.

Availability

Available in Mac OS X v10.3 and later.

Related Sample Code

MovieVideoChart

Declared In

`Movies.h`

GetMediaCreationTime

Returns the creation date and time stored in a media.

```
unsigned long GetMediaCreationTime (
   Media theMedia
);
```

Parameters

theMedia

The media for this operation. Your application obtains this media identifier from such functions as `NewTrackMedia` (page 114) and `GetTrackMedia` (page 96). See `Media Identifiers`.

Return Value

The media's creation date and time.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Movies.h`

GetMediaDataHandler

Determines a media's data handler.

```
DataHandler GetMediaDataHandler (
   Media theMedia,
   short index
);
```

Parameters

theMedia

The media for this operation. Your application obtains this media identifier from such functions as `NewTrackMedia` (page 114) and `GetTrackMedia` (page 96). See `Media Identifiers`.

index

Identifies the data reference. You provide the index value that corresponds to the data reference for which you want to retrieve the data handler. You must set this parameter to 1.

Return Value

A data handler component instance.

Special Considerations

QuickTime normally takes care of selecting data handlers for media. Your application should not need to call this function.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Movies.h`

GetMediaDataHandlerDescription

Retrieves information about a media's data handler.

```
void GetMediaDataHandlerDescription (
   Media theMedia,
   short index,
   OSType *dhType,
   Str255 creatorName,
   OSType *creatorManufacturer
);
```

Parameters

theMedia

The media for this operation. Your application obtains this media identifier from such functions as `NewTrackMedia` (page 114) and `GetTrackMedia` (page 96). See `Media Identifiers`.

index

Identifies the data reference. You provide the index value that corresponds to the data reference for which you want to retrieve the data handler description. You must set this parameter to 1.

dhType

A pointer to a field of data type `OSType`. The Movie Toolbox returns the `data` handler type identifier. This value indicates the type of data reference supported by this data handler. This value also corresponds to the `component` subtype specified for the data handler component. All QuickTime data references have a type value of `'alis'`. If you don't want to receive this information, set this parameter to `NIL`.

creatorName

Points to a string. The Movie Toolbox returns the name of the data handler's creator. If you don't want to receive this information, set this parameter to `NIL`.

creatorManufacturer

A pointer to a long integer. The Movie Toolbox returns the 4-byte value that identifies the manufacturer of the component. If you don't want to retrieve this information, set this parameter to `NIL`.

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

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`Movies.h`

GetMediaDataSize

Determines the size, in bytes, of the sample data in a media segment.

```
long GetMediaDataSize (
   Media theMedia,
   TimeValue startTime,
   TimeValue duration
);
```

Parameters

theMedia

The media for this operation. Your application obtains this media identifier from such functions as `NewTrackMedia` (page 114) and `GetTrackMedia` (page 96). See `Media Identifiers`.

startTime

A time value specifying the starting point of the segment.

duration

A time value that specifies the duration of the segment.

Return Value
The size, in bytes, of the sample data in the defined media segment.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
Movies.h

GetMediaDataSize64

Provides a 64-bit version of GetMediaDataSize.

```
OSErr GetMediaDataSize64 (
   Media theMedia,
   TimeValue startTime,
   TimeValue duration,
   wide *dataSize
);
```

Parameters

theMedia
The media for this operation. Your application obtains this media identifier from such functions as NewTrackMedia (page 114) and GetTrackMedia (page 96). See Media Identifiers.

startTime
A time value specifying the starting point of the segment.

duration
A time value that specifies the duration of the segment.

dataSize
The size, in bytes, of the sample data in the defined media segment.

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

Discussion
The only difference between this function and GetMediaDataSize (page 56) is that the dataSize parameter returns a 64-bit integer instead of the function returning a 32-bit integer.

Special Considerations

New applications should use this function instead of the 32-bit version.

Version Notes
Introduced in QuickTime 4.

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

Declared In
Movies.h

GetMediaDataSizeTime64

Determines the size, in bytes, of the sample data in a media segment.

```
OSErr GetMediaDataSizeTime64 (
   Media theMedia,
   TimeValue64 startDisplayTime,
   TimeValue64 displayDuration,
   SInt64 *dataSize
);
```

Parameters

theMedia

The media for this operation. You obtain this media identifier from such functions as `NewTrackMedia` (page 114) and `GetTrackMedia` (page 96).

startDisplayTime

A 64-bit time value that specifies the starting point of the segment in media display time.

displayDuration

A 64-bit time value that specifies the duration of the segment in media display time.

dataSize

A pointer to a variable to receive the size, in bytes, of the sample data in the defined media segment.

Return Value

An error code. Returns `noErr` if there is no error. You can access Movie Toolbox error returns through `GetMoviesError` and `GetMoviesStickyError`, as well as in the function result.

Discussion

The only difference between this function and GetMediaDataSize64 is that this function uses 64-bit time values and returns a 64-bit size.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`Movies.h`

GetMediaDecodeDuration

Returns the decode duration of a media.

```
TimeValue64 GetMediaDecodeDuration (
   Media theMedia
);
```

Parameters

theMedia

The media for this operation. You obtain this media identifier from such functions as `NewTrackMedia` (page 114) and `GetTrackMedia` (page 96).

Return Value

A 64-bit time value that represents the media's decode duration. A media's decode duration is the sum of the decode durations of its samples.

Availability

Available in Mac OS X v10.3 and later.

Related Sample Code

MovieVideoChart

Declared In
Movies.h

GetMediaDisplayDuration

Returns the display duration of a media.

```
TimeValue64 GetMediaDisplayDuration (
   Media theMedia
);
```

Parameters

theMedia

The media for this operation. You obtain this media identifier from such functions as NewTrackMedia (page 114) and GetTrackMedia (page 96).

Return Value
A 64-bit time value that represents the media's display duration. A media's display duration is its display end time minus its display start time. For media without nonzero display offsets, the decode duration and display duration are the same.

Discussion
When inserting media with display offsets into a track, use display time:

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

Related Sample Code
CaptureAndCompressIPBMovie
MovieVideoChart
OpenGLCaptureToMovie
Quartz Composer QCTV

Declared In
Movies.h

GetMediaDisplayEndTime

Returns the display end time of a media.

```
TimeValue64 GetMediaDisplayEndTime (
   Media theMedia
);
```

Parameters

theMedia

The media for this operation. You obtain this media identifier from such functions as NewTrackMedia (page 114) and GetTrackMedia (page 96).

Return Value
A 64-bit time value that represents the media's display end time. A media's display end time is the sum of the display time and decode duration of the sample with the greatest display time. For media without nonzero display offsets, the display end time is the same as the media's decode duration.

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

Declared In
`Movies.h`

GetMediaDisplayStartTime

Returns the display start time of a media.

```
TimeValue64 GetMediaDisplayStartTime (
   Media theMedia
);
```

Parameters

theMedia

The media for this operation. You obtain this media identifier from such functions as `NewTrackMedia` (page 114) and `GetTrackMedia` (page 96).

Return Value
A 64-bit time value that represents the media's display start time. A media's display start time is the earliest display time of any of its samples. For media without nonzero display offsets, the display start time is always 0.

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

Declared In
`Movies.h`

GetMediaDuration

Returns the duration of a media.

```
TimeValue GetMediaDuration (
   Media theMedia
);
```

Parameters

theMedia

The media for this operation. Your application obtains this media identifier from such functions as `NewTrackMedia` (page 114) and `GetTrackMedia` (page 96). See `Media Identifiers`.

Return Value
The media's duration.

Discussion
The following code sample illustrates the use of `GetMediaDuration`:

```
// GetMediaDuration coding example
// See "Discovering QuickTime," page 89
void CreateMyVideoTrack (Movie movie)
{
    Track   track;
    Media   media;
```

```
    Rect    rect ={0, 0, 100, 320};
    track =NewMovieTrack(movie,
                FixRatio(rect.right, 1),
                FixRatio(rect.bottom, 1),
                kNoVolume);
    media =NewTrackMedia(track,
                VideoMediaType,
                600,                            // video time scale
                NIL, NIL);
    BeginMediaEdits(media);
    MyAddVideoSamplesToMedia(media, &rect);     // assemble data
    EndMediaEdits(media);
    InsertMediaIntoTrack(track,
                0,                              // track start time
                0,                              // media start time
                GetMediaDuration(media),
                kFix1);                         // normal speed
}
```

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
qteffects
qteffects.win
qtshoweffect
qtshoweffect.win
vrmakepano

Declared In
Movies.h

GetMediaHandler

Obtains a reference to a media handler component.

```
MediaHandler GetMediaHandler (
   Media theMedia
);
```

Parameters

theMedia

The media for this operation. Your application obtains this media identifier from such functions as `NewTrackMedia` (page 114) and `GetTrackMedia` (page 96). See `Media Identifiers`.

Return Value
A media handler component instance.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
qttext
qttext.win
vrmovies.win
vrscript
vrscript.win

Declared In
`Movies.h`

GetMediaHandlerDescription

Retrieves information about a media handler.

```
void GetMediaHandlerDescription (
   Media theMedia,
   OSType *mediaType,
   Str255 creatorName,
   OSType *creatorManufacturer
);
```

Parameters

theMedia
The media for this operation. Your application obtains this media identifier from such functions as `NewTrackMedia` (page 114) and `GetTrackMedia` (page 96). See `Media Identifiers`.

mediaType
A pointer to a field in which the Movie Toolbox returns the `media` type identifier (see below). This value indicates the type of media supported by this media handler. This value also corresponds to the `component` subtype specified for the media handler component. If you don't want to receive this information, set the `mediaType` parameter to `NIL`. See these constants:

```
VideoMediaType
SoundMediaType
TextMediaType
```

creatorName
Points to a string. The Movie Toolbox returns the name of the media handler's creator. If you don't want to receive this information, set this parameter to `NIL`.

creatorManufacturer
A pointer to a long integer. The Movie Toolbox returns the 4-byte value that identifies the manufacturer of the component. If you don't want to retrieve this information, set this parameter to `NIL`.

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

Discussion
The following code sample illustrates the use of `GetMediaHandlerDescription`:

```
// GetMediaHandlerDescription coding example
// See "Discovering QuickTime," page 363
```

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

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
CompressMovies
DigitizerShell
DragAndDrop Shell
MovieGWorlds
QT Internals

Declared In
`Movies.h`

GetMediaInputMap

Returns a copy of the input map associated with a specified media.

```
OSErr GetMediaInputMap (
   Media theMedia,
   QTAtomContainer *inputMap
);
```

Parameters

theMedia

The media for this operation. Your application obtains this media identifier from such functions as `NewTrackMedia` (page 114) and `GetTrackMedia` (page 96). See `Media Identifiers`.

inputMap

The media input map for this operation. You must dispose of the map referred to by this parameter when you are done with it using `QTDisposeAtomContainer`.

Return Value

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

Discussion

Use this function to specify the media you want to get so you can modify its input map, as illustrated below:

```
// GetMediaInputMap coding example
// See "Discovering QuickTime," page 365
#define kImageIndexToOverride          1
Movie               movie1, movie2;
long                lReferenceIndex, lImageIndexToOverride;
Track               trackSprite;
QTAtomContainer     qtacInputMap;
QTAtom              lInputAtom;
OSType              dwInputType;
Media               mediaSprite;
// get the sprite media's input map
mediaSprite =GetTrackMedia(trackSprite);
GetMediaInputMap(mediaSprite, &qtacInputMap);
// add an atom for a modifier track
QTInsertChild(qtacInputMap, kParentAtomIsContainer, kTrackModifierInput,
               lReferenceIndex, 0, 0, NIL, &lInputAtom);
// add a child atom to specify the input type
dwInputType =kTrackModifierTypeImage;
QTInsertChild(qtacInputMap, lInputAtom, kTrackModifierType, 1, 0,
               sizeof(dwInputType), &dwInputType, NIL);
// add a second child atom to specify index of image to override
lImageIndexToOverride =EndianS16_NtoB(kImageIndexToOverride);
QTInsertChild(qtacInputMap, lInputAtom, kSpritePropertyImageIndex, 1, 0,
        sizeof(lImageIndexToOverride), &lImageIndexToOverride, NIL);
// update the sprite media's input map
SetMediaInputMap(mediaSprite, qtacInputMap);
QTDisposeAtomContainer(qtacInputMap);
```

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

ChromaKeyMovie

Declared In
Movies.h

GetMediaLanguage

Returns a media's localized language or region code.

```
short GetMediaLanguage (
   Media theMedia
);
```

Parameters

theMedia

The media for this operation. Your application obtains this media identifier from such functions as NewTrackMedia (page 114) and GetTrackMedia (page 96). See Media Identifiers.

Return Value
The media's language or region code.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
Movies.h

GetMediaModificationTime

Returns a media's modification date and time.

```
unsigned long GetMediaModificationTime (
   Media theMedia
);
```

Parameters

theMedia

The media for this operation. Your application obtains this media identifier from such functions as NewTrackMedia (page 114) and GetTrackMedia (page 96). See Media Identifiers.

Return Value
The media's modification date and time.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
Movies.h

GetMediaPreferredChunkSize

Retrieves the maximum chunk size for a media.

```
OSErr GetMediaPreferredChunkSize (
   Media theMedia,
   long *maxChunkSize
);
```

Parameters

theMedia

The media for this operation. Your application obtains this media identifier from such functions as NewTrackMedia (page 114) and GetTrackMedia (page 96). See Media Identifiers.

maxChunkSize

Specifies a field to receive the maximum chunk size, in bytes.

Return Value

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

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

Movies.h

GetMediaQuality

Returns a media's quality level value.

```
short GetMediaQuality (
   Media theMedia
);
```

Parameters

theMedia

The media for this operation. Your application obtains this media identifier from such functions as NewTrackMedia (page 114) and GetTrackMedia (page 96). See Media Identifiers.

Return Value

A short integer whose bits indicate quality constants (see below). More than one of these bits may be set to 1.

Discussion

The Movie Toolbox uses this quality value to influence which track of a movie it selects to play on a given computer. This even applies to sound media. The low-order 6 bits specify pixel depths and the upper 2 bits specify quality levels. If a bit is set to 1, the media can be played at the corresponding depth and quality level.

Version Notes

Introduced in QuickTime 3 or earlier.

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

Related Sample Code
vrmovies
vrmovies.win

Declared In
`Movies.h`

GetMediaSample

Returns a sample from a movie data file.

```
OSErr GetMediaSample (
   Media theMedia,
   Handle dataOut,
   long maxSizeToGrow,
   long *size,
   TimeValue time,
   TimeValue *sampleTime,
   TimeValue *durationPerSample,
   SampleDescriptionHandle sampleDescriptionH,
   long *sampleDescriptionIndex,
   long maxNumberOfSample,
   long *numberOfSamples,
   short *sampleFlags
);
```

Parameters

theMedia

The media for this operation. Your application obtains this media identifier from such functions as `NewTrackMedia` (page 114) and `GetTrackMedia` (page 96). See `Media Identifiers`.

dataOut

A handle. The `GetMediaSample` function returns the sample data into this handle. The function increases the size of this handle, if necessary. You can specify the handle's maximum size with the `maxSizeToGrow` parameter.

maxSizeToGrow

The maximum number of bytes of sample data to be returned. The `GetMediaSample` function does not increase the handle specified by the `dataOut` parameter to a size greater than you specify with this parameter. Set this value to 0 to enforce no limit on the number of bytes to be returned.

size

A pointer to a long integer. The `GetMediaSample` function updates the field referred to by the `size` parameter with the number of bytes of sample data returned in the handle specified by the `dataOut` parameter. Set this parameter to `NIL` if you are not interested in this information.

time

The starting time of the sample to be retrieved. You must specify this value in the media's time scale.

sampleTime

A pointer to a time value. The `GetMediaSample` function updates this time value to indicate the actual time of the returned sample data. (The returned time may differ from the time you specified with the `time` parameter. This will occur if the time you specified falls in the middle of a sample.) If you are not interested in this information, set this parameter to `NIL`.

durationPerSample

A pointer to a time value. The Movie Toolbox returns the duration of each sample in the media. This time value is expressed in the media's time scale. Set this parameter to 0 if you don't want this information.

sampleDescriptionH

A handle to a `SampleDescription` structure. The `GetMediaSample` function returns the sample description corresponding to the returned sample data. The function resizes this handle as appropriate. If you don't want a `SampleDescription` structure, set this parameter to `NIL`.

sampleDescriptionIndex

A pointer to a long integer. The `GetMediaSample` function returns an index value to the `SampleDescription` structure that corresponds to the returned sample data. You can retrieve the structure by calling `GetMediaSampleDescription` (page 71) and passing this index in the `descH` parameter. If you don't want this information, set this parameter to `NIL`.

maxNumberOfSamples

The maximum number of samples to be returned. The Movie Toolbox does not return more samples than you specify with this parameter. If you set this parameter to 0, the Movie Toolbox uses a value that is appropriate for the media, and returns that value in the field referenced by the `numberOfSamples` parameter.

numberOfSamples

A pointer to a long integer. The `GetMediaSample` function updates the field referred to by this parameter with the number of samples it actually returns. If you don't want this information, set this parameter to `NIL`.

sampleFlags

A pointer to a short integer in which `GetMediaSample` returns flags (see below) that describe the sample. Unused flags are set to 0. If you don't want this information, set this parameter to `NIL`. See these constants:

```
mediaSampleNotSync
```

Return Value

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

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

qttext.win

qtwiredactions

SoundPlayer.win

vrmakepano

vrmakepano.win

Declared In
Movies.h

GetMediaSample2

Retrieves sample data from a media file.

```
OSErr GetMediaSample2 (
   Media theMedia,
   UInt8 *dataOut,
   ByteCount maxDataSize,
   ByteCount *size,
   TimeValue64 decodeTime,
   TimeValue64 *sampleDecodeTime,
   TimeValue64 *decodeDurationPerSample,
   TimeValue64 *displayOffset,
   SampleDescriptionHandle sampleDescriptionH,
   ItemCount *sampleDescriptionIndex,
   ItemCount maxNumberOfSamples,
   ItemCount *numberOfSamples,
   MediaSampleFlags *sampleFlags
);
```

Parameters

theMedia

The media for this operation. You obtain this media identifier from such functions as NewTrackMedia (page 114) and GetTrackMedia (page 96).

dataOut

A pointer to a buffer to receive sample data. The buffer must be large enough to contain at least maxDataSize bytes. If you do not want to receive sample data, pass NULL.

maxDataSize

The maximum number of bytes allocated to hold the sample data.

size

A pointer to memory where the function returns the number of bytes of sample data returned in the memory area specified by dataOut. Set this parameter to NULL if you are not interested in this information.

decodeTime

The starting time of the sample to be retrieved in decode time. You must specify this value in the media's time scale.

sampleDecodeTime

A pointer to a time value in decode time. The function updates this time value to indicate the actual time of the returned sample data. (The returned time may differ from the time you specified with the time parameter. This will occur if the time you specified falls in the middle of a sample.) If you are not interested in this information, set this parameter to NULL.

decodeDurationPerSample

A pointer to a time value in decode time. The Movie Toolbox returns the duration of each sample in the media. Set this parameter to NULL if you don't want this information.

displayOffset

A pointer to a time value. The function updates this time value to indicate the display offset of the returned sample. This time value is expressed in the media's time scale. Set this parameter to NULL if you don't want this information.

sampleDescriptionH

A handle to a SampleDescription structure. The function returns the sample description corresponding to the returned sample data. The function resizes this handle as appropriate. If you don't want a SampleDescription structure, set this parameter to NIL.

sampleDescriptionIndex

A pointer to a long integer. The function returns an index value to the SampleDescription structure that corresponds to the returned sample data. You can retrieve the structure by calling GetMediaSampleDescription (page 71) and passing this index in the descH parameter. If you don't want this information, set this parameter to NIL.

maxNumberOfSamples

The maximum number of samples to be returned. The Movie Toolbox does not return more samples than you specify with this parameter. If you set this parameter to 0, the Movie Toolbox uses a value that is appropriate for the media, and returns that value in the field referenced by the numberOfSamples parameter.

numberOfSamples

A pointer to a long integer. The function updates the field referred to by this parameter with the number of samples it actually returns. If you don't want this information, set this parameter to NULL.

sampleFlags

A pointer to a short integer in which the function returns flags that describe the sample. Unused flags are set to 0. If you don't want this information, set this parameter to NULL: mediaSampleNotSync This flag is set to 1 if the sample is not a sync sample and to 0 if the sample is a sync sample. See these constants:

mediaSampleNotSync

Return Value
You can access this function's error returns through GetMoviesError and GetMoviesStickyError. It returns paramErr if there is a bad parameter value, maxSizeToGrowTooSmall if the sample data is larger than maxDataSize, or noErr if there is no error.

Discussion
Whereas GetMediaSample (page 67) takes a resizable Handle and a maxSizeToGrow parameter, GetMediaSample2 takes a pointer and a maxDataSize parameter. If you want to read a sample into a Handle, you can use the following code:

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

Related Sample Code
MovieVideoChart

Declared In
Movies.h

GetMediaSampleCount

Determines the number of samples in a media.

```
long GetMediaSampleCount (
   Media theMedia
);
```

Parameters

theMedia

The media for this operation. You obtain this media identifier from such functions as `NewTrackMedia` (page 114) and `GetTrackMedia` (page 96).

Return Value
The number of samples in the media.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
MovieVideoChart
vrmakepano
vrmakepano.win

Declared In
`Movies.h`

GetMediaSampleDescription

Retrieves a SampleDescription structure from a media.

```
void GetMediaSampleDescription (
   Media theMedia,
   long index,
   SampleDescriptionHandle descH
);
```

Parameters

theMedia

The media for this operation. You obtain this media identifier from such functions as `NewTrackMedia` (page 114) and `GetTrackMedia` (page 96).

index

The index of the `SampleDescription` structure to retrieve. This index corresponds to the structure itself, not to the samples in the media. Index numbers start with 1.

descH

Specifies a handle that is to receive the `SampleDescription` structure. The Movie Toolbox correctly resizes this handle for the returned structure. If there is no description for the specified index, the function returns this handle unchanged. Your application must allocate and dispose of this handle.

Discussion
The Movie Toolbox identifies a media's sample descriptions with an index value, ranging from 1 to the number of sample descriptions in the media. Sample description indexes provide a convenient way to access each sample description in a media. You can access error returns from this function through `GetMoviesError` and `GetMoviesStickyError`. See `Error Codes`.

Special Considerations

The format of sample descriptions differs by media type. Sample descriptions for image data are defined by `ImageDescription` structures. Sample descriptions for sound are defined by `SoundDescription` structures. Sample descriptions for text are defined by `TextDescription` structures.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
CompressMovies
DigitizerShell
DragAndDrop Shell
MovieGWorlds
QT Internals

Declared In
`Movies.h`

GetMediaSampleDescriptionCount

Returns the number of sample descriptions in a media.

```
long GetMediaSampleDescriptionCount (
   Media theMedia
);
```

Parameters

`theMedia`

The media for this operation. You obtain this media identifier from such functions as `NewTrackMedia` (page 114) and `GetTrackMedia` (page 96).

Return Value
The number of sample descriptions in the media.

Special Considerations

The format of sample descriptions differs by media type. Sample descriptions for image data are defined by `ImageDescription` structures. Sample descriptions for sound are defined by `SoundDescription` structures. Sample descriptions for text are defined by `TextDescription` structures.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`Movies.h`

GetMediaSampleReference

Obtains reference information about samples that are stored in a movie data file.

```
OSErr GetMediaSampleReference (
   Media theMedia,
   long *dataOffset,
   long *size,
   TimeValue time,
   TimeValue *sampleTime,
   TimeValue *durationPerSample,
   SampleDescriptionHandle sampleDescriptionH,
   long *sampleDescriptionIndex,
   long maxNumberOfSamples,
   long *numberOfSamples,
   short *sampleFlags
);
```

Parameters

theMedia

The media for this operation. Your application obtains this media identifier from such functions as `NewTrackMedia` (page 114) and `GetTrackMedia` (page 96). See `Media Identifiers`.

dataOffset

A pointer to a long integer. `GetMediaSampleReference` updates the field referred to by this parameter with the offset to the sample data. This parameter is used differently by each media handler. For example, the hierarchical file system (HFS) media handler returns an offset into the file that contains the media data.

size

A pointer to a long integer. `GetMediaSampleReference` updates the field referred to by the `size` parameter with the number of bytes of sample data referred to by the reference. Set this parameter to `NIL` if you are not interested in this information.

time

The starting time of the sample reference to be retrieved. You must specify this value in the media's time scale.

sampleTime

A pointer to a time value. `GetMediaSampleReference` updates this time value to indicate the actual time of the returned sample data. (The returned time may differ from the time you specified with the `time` parameter. This will occur if the time you specified falls in the middle of a sample.) If you are not interested in this information, set this parameter to `NIL`.

durationPerSample

A pointer to a time value. The Movie Toolbox returns the duration of each sample in the media. This time value is expressed in the media's time scale. Set this parameter to 0 if you don't want this information.

sampleDescriptionH

A handle to a `SampleDescription` structure. `GetMediaSampleReference` returns the structure corresponding to the returned sample data. The function resizes this handle as appropriate. If you don't want the `SampleDescription` structure, set this parameter to `NIL`.

sampleDescriptionIndex

A pointer to a long integer. `GetMediaSampleReference` returns an index value to the `SampleDescription` structure that corresponds to the returned sample data. To retrieve the media sample description, pass this index in the `descH` parameter of `GetMediaSampleDescription` (page 71). If you don't want this information, set this parameter to `NIL`.

maxNumberOfSamples

The maximum number of samples to be returned. The Movie Toolbox does not return a reference that refers to more samples than you specify with this parameter. If you set this parameter to 0, the Movie Toolbox uses a value that is appropriate for the media and returns that value in the field referenced by the `numberOfSamples` parameter.

numberOfSamples

A pointer to a long integer. `GetMediaSampleReference` updates the field referred to by this parameter with the number of samples referred to by the returned reference. If you don't want this information, set this parameter to `NIL`.

sampleFlags

A pointer to a short integer in which `GetMediaSampleReference` returns flags (see below) that describe the samples referred to by the returned reference. Unused flags are set to 0. If you don't want this information, set this parameter to `NIL`. See these constants:

```
mediaSampleNotSync
```

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

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
SlideShowImporter
SlideShowImporter.win

Declared In
`Movies.h`

GetMediaSampleReferences

Obtains reference information about groups of samples that are stored in a movie.

```
OSErr GetMediaSampleReferences (
   Media theMedia,
   TimeValue time,
   TimeValue *sampleTime,
   SampleDescriptionHandle sampleDescriptionH,
   long *sampleDescriptionIndex,
   long maxNumberOfEntries,
   long *actualNumberofEntries,
   SampleReferencePtr sampleRefs
);
```

Parameters

theMedia

The media for this operation. Your application obtains this media identifier from such functions as `NewTrackMedia` (page 114) and `GetTrackMedia` (page 96). See `Media Identifiers`.

time

The starting time of the sample references to be retrieved. You must specify this value in the media's time scale.

sampleTime

A pointer to a time value. `GetMediaSampleReferences` updates this time value to indicate the actual time of the first returned sample data. If you are not interested in this information, set this parameter to `NIL`.

sampleDescriptionH

A handle to a `SampleDescription` structure. `GetMediaSampleReference` (page 73) returns the structure corresponding to the returned sample data. The function resizes this handle as appropriate. `GetMediaSampleReferences` only returns a single sample description. If the sample description changes within the media, `GetMediaSampleReferences` returns only as many samples as use a single sample description. You must call it again to get the next group of samples using the next sample description. If you don't want the `SampleDescription` structure, set this parameter to `NIL`.

sampleDescriptionIndex

A pointer to a long integer. `GetMediaSampleReferences` returns an index value to the `SampleDescription` structures that correspond to the returned sample data. Use this index to retrieve the media sample descriptions with `GetMediaSampleDescription` (page 71). If you don't want this information, set this parameter to `NIL`.

maxNumberOfEntries

The maximum number of entries to be returned. The sample references pointer provided by the `sampleRefs` parameter must be large enough to receive the number of entries specified by this parameter. The toolbox does not return more entries than you specify with this parameter. It may, however, return fewer.

actualNumberofEntries

A pointer to a long integer. `GetMediaSampleReferences` updates the field referred to by this parameter with the number of entries referred to by the returned reference.

sampleRefs

A pointer to the number of `SampleReferenceRecord` structures specified in the `maxNumberOfEntries` parameter. On return from this call, the number of sample reference records indicated by the value returned in `actualNumberofEntries` will be filled in.

Return Value

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

Discussion

Using this function instead of `GetMediaSampleReference` (page 73) can greatly increase the performance of operations that need access to information about each sample in a movie. No information is returned from this call that wasn't previously available from `GetMediaSampleReference`.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Movies.h`

GetMediaSampleReferences64

Provides a 64-bit version of GetMediaSampleReferences.

```
OSErr GetMediaSampleReferences64 (
   Media theMedia,
   TimeValue time,
   TimeValue *sampleTime,
   SampleDescriptionHandle sampleDescriptionH,
   long *sampleDescriptionIndex,
   long maxNumberOfEntries,
   long *actualNumberofEntries,
   SampleReference64Ptr sampleRefs
);
```

Parameters

theMedia

The media for this operation. Your application obtains this media identifier from such functions as `NewTrackMedia` (page 114) and `GetTrackMedia` (page 96). See `Media Identifiers`.

time

The starting time of the sample references to be retrieved. You must specify this value in the media's time scale.

sampleTime

A pointer to a time value. GetMediaSampleReferences64 updates this time value to indicate the actual time of the first returned sample data. If you are not interested in this information, set this parameter to `NIL`.

sampleDescriptionH

A handle to a `SampleDescription` structure. `GetMediaSampleReference` (page 73) returns the sample description corresponding to the returned sample data. The function resizes this handle as appropriate. `GetMediaSampleReferences` only returns a single structure. If the sample description changes within the media, `GetMediaSampleReferences` returns only as many samples as use a single sample description. You must call it again to get the next group of samples using the next sample description. If you don't want the `SampleDescription` structure, set this parameter to `NIL`.

sampleDescriptionIndex

A pointer to a long integer. GetMediaSampleReferences64 returns an index value to the sample descriptions that correspond to the returned sample data. Use this index to retrieve the media sample descriptions with `GetMediaSampleDescription` (page 71). If you don't want this information, set this parameter to `NIL`.

maxNumberOfEntries

The maximum number of entries to be returned. The sample references pointer provided by the `sampleRefs` parameter must be large enough to receive the number of entries specified by this parameter. The toolbox does not return more entries than you specify with this parameter. It may, however, return fewer.

actualNumberofEntries

A pointer to a long integer. GetMediaSampleReferences64 updates the field referred to by this parameter with the number of entries referred to by the returned reference.

sampleRefs

A pointer to the number of `SampleReference64Record` structures specified in the `maxNumberOfEntries` parameter. On return from this call, the number of sample reference records indicated by the value returned in `actualNumberofEntries` will be filled in.

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

Discussion
The only difference between this function and `GetMediaSampleReferences` (page 74) is that the `sampleRefs` parameter points to SampleReference64Record structures instead of `SampleReferenceRecord` structures.

Special Considerations

New applications should use this function instead of the 32-bit version.

Version Notes
Introduced in QuickTime 4.

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

Declared In
`Movies.h`

GetMediaShadowSync

Obsolete; no longer supported.

```
OSErr GetMediaShadowSync (
   Media theMedia,
   long frameDiffSampleNum,
   long *syncSampleNum
);
```

Parameters

`theMedia`

Indicates the media in which the shadow sync sample has been established and from which the shadow sync number is to be obtained.

`frameDiffSampleNum`

The frame difference sample number associated with the desired shadow sync sample number.

`syncSampleNum`

A pointer to the sample number of the shadow sync sample. If the media does not have a shadow sync sample, 0 is returned in the `syncSampleNum` parameter.

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

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`Movies.h`

GetMediaSyncSampleCount

Gets the number of sync samples in a media.

```
long GetMediaSyncSampleCount (
   Media theMedia
);
```

Parameters

theMedia

The media for this operation. Your application obtains this media identifier from such functions as `NewTrackMedia` (page 114) and `GetTrackMedia` (page 96). See `Media Identifiers`.

Return Value

The number of sync samples in the media.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Movies.h`

GetMediaTimeScale

Determines a media's time scale.

```
TimeScale GetMediaTimeScale (
   Media theMedia
);
```

Parameters

theMedia

The media for this operation. Your application obtains this media identifier from such functions as `NewTrackMedia` (page 114) and `GetTrackMedia` (page 96). See `Media Identifiers`.

Return Value

The media's time scale.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

AddFrameToMovie

BurntTextSampleCode

MovieVideoChart

qttext

qttext.win

Declared In
Movies.h

GetMediaTrack

Determines the track that uses a specified media.

```
Track GetMediaTrack (
   Media theMedia
);
```

Parameters

theMedia

The media for this operation. Your application obtains this media identifier from such functions as NewTrackMedia (page 114) and GetTrackMedia (page 96). See Media Identifiers.

Return Value
The track identifier of the track that uses the specified media.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
CaptureAndCompressIPBMovie
QTExtractAndConvertToMovieFile
qtmovietrack
qtmovietrack.win
SCAudioCompress

Declared In
Movies.h

GetMediaUserData

Obtains access to a media's user data list.

```
UserData GetMediaUserData (
   Media theMedia
);
```

Parameters

theMedia

The media for this operation. Your application obtains this media identifier from such functions as NewTrackMedia (page 114) and GetTrackMedia (page 96). See Media Identifiers.

Return Value
The media's user data list.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`Movies.h`

GetMovieDataSize

Determines the size of the sample data in a segment of a movie.

```
long GetMovieDataSize (
   Movie theMovie,
   TimeValue startTime,
   TimeValue duration
);
```

Parameters

`theMovie`

The movie for this operation. You obtain this movie identifier from such functions as `NewMovie`, `NewMovieFromFile`, and `NewMovieFromHandle`.

`startTime`

A time value specifying the starting point of the segment.

`duration`

A time value that specifies the duration of the segment.

Return Value
The size, in bytes, of the sample data in the defined segment of the designated movie.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`Movies.h`

GetMovieDataSize64

Provides a 64-bit version of GetMovieDataSize.

```
OSErr GetMovieDataSize64 (
   Movie theMovie,
   TimeValue startTime,
   TimeValue duration,
   wide *dataSize
);
```

Parameters

`theMovie`

The movie for this operation. You obtain this movie identifier from such functions as `NewMovie`, `NewMovieFromFile`, and `NewMovieFromHandle`.

startTime
A time value specifying the starting point of the segment.

duration
A time value that specifies the duration of the segment.

data size
The size, in bytes, of the sample data in the defined segment of the designated movie.

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

Discussion
The only difference between this function and `GetMovieDataSize` (page 80) is that the `dataSize` parameter is a 64-bit integer instead of a 32-bit integer.

Special Considerations
New applications should use this function instead of the 32-bit version.

Version Notes
Introduced in QuickTime 4.

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

Declared In
`Movies.h`

GetMovieImporterForDataRef

Gets the movie importer component for a movie.

```
OSErr GetMovieImporterForDataRef (
   OSType dataRefType,
   Handle dataRef,
   long flags,
   Component *importer
);
```

Parameters
dataRefType
The type of data reference; see `Data References`.

dataRef
A handle to the `data` reference. The type of information stored in the handle depends upon the `data` reference type specified by `dataRefType`.

flags
Flags (see below) that modify this function's behavior. See these constants:
`kGetMovieImporterDontConsiderGraphicsImporters`

importer
A pointer to an importer component that can import the movie. Returns `NIL` if no importer can be found.

Return Value
If this function is allowed to use `async` calls (by being passed `kGetMovieImporterUseAsyncCalls` in the `flags` parameter), it returns `notEnoughDataErr` if it would block. You can access this error return through `GetMoviesError` and `GetMoviesStickyError`, as well as in the function result. For other errors, see `Error Codes`.

Discussion
You can use `GetMovieImporterForDataRef` to determine if a file can be opened by QuickTime as a movie (for example, in a drag-and-drop operation) as illustrated below:

```
AliasHandle            alias;
MovieImportComponent   mi;
NewAliasMinimal(&reply.sfFile, &alias);
GetMovieImporterForDataRef(rAliasType, (Handle)alias,
kGetMovieImporterDontConsiderGraphicsImporters, &mi);
DisposeHandle((Handle)alias);
if (mi !=NIL) {
    // this file can be opened as a movie
    . . .
    }
```

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
MakeEffectMovie
Movie From DataRef
qteffects.win
vrbackbuffer
vrmovies.win

Declared In
`Movies.h`

GetMovieIndTrack

Determines the track identifier of a track, given the track's index value.

```
Track GetMovieIndTrack (
   Movie theMovie,
   long index
);
```

Parameters

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

index
The index value of the track for this operation.

Return Value
A track identifier. If the function cannot locate the track, it sets this returned value to `NIL`.

Discussion
This function returns the track identifier that is appropriate to the specified track. The index value identifies the track among all current tracks in a movie. Index values range from 1 to the number of tracks in the movie. The following code sample illustrates its use:

```
// GetMovieIndTrack coding example
// See "Discovering QuickTime," page 363
Movie           movie1;
TimeValue       lOldDuration;
Movie           movie2;
long            lIndex, lOrigTrackCount, lReferenceIndex;
Track           track, trackSprite;
// get the first track in original movie and position at the start
trackSprite =GetMovieIndTrack(movie1, 1);
SetMovieSelection(movie1, 0, 0);
// remove all tracks except video in modifier movie
for (lIndex =1; lIndex <=GetMovieTrackCount(movie2); lIndex++) {
    Track       track =GetMovieIndTrack(movie2, lIndex);
    OSType      dwType;
    GetMediaHandlerDescription(GetTrackMedia(track),
                                &dwType, NIL, NIL);
    if (dwType !=VideoMediaType) {
        DisposeMovieTrack(track);
        lIndex--;
    }
}
// add the modifier track to original movie
lOldDuration =GetMovieDuration(movie1);
AddMovieSelection(movie1, movie2);
DisposeMovie(movie2);
// truncate the movie to the length of the original track
DeleteMovieSegment(movie1, lOldDuration,
                   GetMovieDuration(movie1) - lOldDuration);
// associate the modifier track with the original sprite track
track =GetMovieIndTrack(movie1, lOrigTrackCount + 1);
AddTrackReference(trackSprite, track, kTrackModifierReference,
                   &lReferenceIndex);
```

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
ChromaKeyMovie
QT Internals
qtinfo
qtinfo.win
qttext.win

Declared In
`Movies.h`

GetMovieIndTrackType

Searches for all of a movie's tracks that share a given media type or media characteristic.

```
Track GetMovieIndTrackType (
   Movie theMovie,
   long index,
   OSType trackType,
   long flags
);
```

Parameters

theMovie

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

index

The index value of the track for this operation. This is not that same as the track's index value in the movie. Rather, this parameter is an index into the set of tracks that meet your other selection criteria.

trackType

Contains either a media type or a media characteristic value. The toolbox applies this value to the search, and returns information about tracks that meet this criterion. You indicate whether you have specified a media type or characteristic value by setting the `flags` parameter appropriately.

flags

Contains flags (see below) that control the search operation. Note that you may not set both `movieTrackMediaType` and `movieTrackCharacteristic` to 1. See these constants:

```
movieTrackMediaType
movieTrackCharacteristic
movieTrackEnabledOnly
```

Return Value

A track identifier.

Discussion

The toolbox returns the track identifier that corresponds to the track that meets your selection criteria. If the toolbox cannot find a matching track, in returns a value of `NIL`. Note that the `index` parameter does not work the same way that is does in `GetMovieIndTrack` (page 82). With `GetMovieIndTrackType`, the `index` parameter specifies an index into the set of tracks that meet your other selection criteria. For example, in order to find the third track that supports the sound characteristic, you would call the function in the following manner:

```
theTrack =GetMovieIndTrackType (theMovie, 3, AudioMediaCharacteristic,
movieTrackCharacteristic);
```

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

MakeEffectMovie

qteffects.win

qttext

qttext.win
qttimecode.win

Declared In
`Movies.h`

GetMovieTrack

Determines the track identifier of a track, given the track's ID value.

```
Track GetMovieTrack (
   Movie theMovie,
   long trackID
);
```

Parameters

`theMovie`

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

`trackID`

The ID value of the track for this operation.

Return Value
A track identifier.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`Movies.h`

GetMovieTrackCount

Returns the number of tracks in a movie.

```
long GetMovieTrackCount (
   Movie theMovie
);
```

Parameters

`theMovie`

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

Return Value
The number of tracks in the movie.

Discussion
The following code sample illustrates the use of `GetMovieTrackCount`:

```
// GetMovieTrackCount coding example
```

```
// See "Discovering QuickTime," page 363
Movie           movie1;
TimeValue       lOldDuration;
Movie           movie2;
long            lIndex, lOrigTrackCount, lReferenceIndex;
Track           track, trackSprite;
// get the first track in original movie and position at the start
trackSprite =GetMovieIndTrack(movie1, 1);
SetMovieSelection(movie1, 0, 0);
// remove all tracks except video in modifier movie
for (lIndex =1; lIndex <=GetMovieTrackCount(movie2); lIndex++) {
    Track       track =GetMovieIndTrack(movie2, lIndex);
    OSType      dwType;
    GetMediaHandlerDescription(GetTrackMedia(track),
                               &dwType, NIL, NIL);
    if (dwType !=VideoMediaType) {
        DisposeMovieTrack(track);
        lIndex--;
    }
}
// add the modifier track to original movie
lOldDuration =GetMovieDuration(movie1);
AddMovieSelection(movie1, movie2);
DisposeMovie(movie2);
// truncate the movie to the length of the original track
DeleteMovieSegment(movie1, lOldDuration,
                   GetMovieDuration(movie1) - lOldDuration);
// associate the modifier track with the original sprite track
track =GetMovieIndTrack(movie1, lOrigTrackCount + 1);
AddTrackReference(trackSprite, track, kTrackModifierReference,
                   &lReferenceIndex);
```

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
QT Internals
qtinfo
qtinfo.win
qttext
qttext.win

Declared In
`Movies.h`

GetNextTrackReferenceType

Determines all of the track reference types that are defined for a given track.

```
OSType GetNextTrackReferenceType (
   Track theTrack,
   OSType refType
);
```

Parameters

theTrack

Identifies the track for this operation. Your application obtains this track identifier from such functions as `NewMovieTrack` (page 112) and `GetMovieTrack` (page 85).

refType

The type of reference. Set this parameter to 0 to retrieve the first track reference type. On subsequent requests, use the previous value returned by this function.

Return Value

An `OSType` containing the next track reference type value defined for the track; see `Data References`.

Discussion

There is no implied ordering of the values returned by this function . When you reach the end of the track's reference types, this function sets the returned value to 0.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

MakeEffectMovie

Movie From DataRef

qteffects.win

QTKitTimeCode

vrmovies.win

Declared In

`Movies.h`

GetTrackAlternate

Determines all the tracks in an alternate group.

```
Track GetTrackAlternate (
   Track theTrack
);
```

Parameters

theTrack

The track for this operation. Your application obtains this track identifier from such functions as `NewMovieTrack` (page 112) and `GetMovieTrack` (page 85).

Return Value

The track identifier of the next track in the group.

Discussion
This function returns the track identifier of the next track in the group. Because the alternate group list is circular, you must specify a different track in the group each time you call this function. You have retrieved all the tracks in the group when the function returns the track identifier that you supplied the first time you called `GetTrackAlternate`. If there is only one track in an alternate group, or if the track you specify does not belong to a group, this function returns the track identifier you supply.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`Movies.h`

GetTrackCreationTime

Returns a track's creation date and time.

```
unsigned long GetTrackCreationTime (
   Track theTrack
);
```

Parameters

theTrack

The track for this operation. Your application obtains this track identifier from such functions as `NewMovieTrack` (page 112) and `GetMovieTrack` (page 85).

Return Value
The track's creation date and time.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`Movies.h`

GetTrackDataSize

Determines the size, in bytes, of the sample data in a segment of a track.

```
long GetTrackDataSize (
   Track theTrack,
   TimeValue startTime,
   TimeValue duration
);
```

Parameters

theTrack

The track for this operation. You obtain this track identifier from such functions as `NewMovieTrack` (page 112) and `GetMovieTrack` (page 85).

startTime

A time value specifying the starting point of the segment.

duration

A time value that specifies the duration of the segment.

Return Value

The size, in bytes, of the sample data in a segment of a track.

Discussion

This function counts each use of a sample. That is, if a track uses a given sample more than once, the size of that sample is included in the returned size value one time for each use. Consequently, the returned size is greater than or equal to the actual size of the track's sample data.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Movies.h`

GetTrackDataSize64

Provides a 64-bit version of GetTrackDataSize.

```
OSErr GetTrackDataSize64 (
   Track theTrack,
   TimeValue startTime,
   TimeValue duration,
   wide *dataSize
);
```

Parameters

theTrack

A track identifier. Your application obtains this identifier from such functions as `NewMovieTrack` (page 112) and `GetMovieTrack` (page 85).

startTime

A time value specifying the starting point of the segment.

duration

A time value that specifies the duration of the segment.

dataSize

The size, in bytes, of the sample data in a segment of a track. This function counts each use of a sample. That is, if a track uses a given sample more than once, the size of that sample is included in the returned size value one time for each use. Consequently, the returned size is greater than or equal to the actual size of the track's sample data.

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

Discussion
The only difference between this function and `GetTrackDataSize` (page 88) is that size of the sample data is returned as a 64-bit integer in the `dataSize` parameter instead of as a 32-bit integer returned by the function.

Special Considerations

New applications should use this function instead of the 32-bit version.

Version Notes
Introduced in QuickTime 4.

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

Declared In
`Movies.h`

GetTrackDimensions

Determines a track's source rectangle.

```
void GetTrackDimensions (
   Track theTrack,
   Fixed *width,
   Fixed *height
);
```

Parameters

theTrack

The track for this operation. Your application obtains this track identifier from such functions as `NewMovieTrack` and `GetMovieTrack` (page 85).

width

A pointer to a fixed-point number. The Movie Toolbox returns the width, in pixels, of the track's rectangle. This value corresponds to the x coordinate of the lower-right corner of the track's rectangle.

height

A pointer to a fixed-point number. The Movie Toolbox returns the height, in pixels, of the track's rectangle. This value corresponds to the y coordinate of the lower-right corner of the track's rectangle.

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

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
qteffects.win
qtstreamsplicer.win
qttext
qttext.win
SlideShowImporter

Declared In
`Movies.h`

GetTrackDisplayMatrix

Returns a matrix that is the concatenation of all matrices currently affecting the track's location, scaling, and so on, including the movie's matrix, the track's matrix, and the modifier matrix.

```
OSErr GetTrackDisplayMatrix (
   Track theTrack,
   MatrixRecord *matrix
);
```

Parameters

theTrack

The track for this operation. Your application obtains this track identifier from such functions as `NewMovieTrack` (page 112) and `GetMovieTrack` (page 85).

matrix

A pointer to a matrix structure.

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

Discussion
Since modifier information is passed between tracks at `MoviesTask` time, the information returned by this call represents the matrix in effect at the last `MoviesTask` call.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`Movies.h`

GetTrackDuration

Returns the duration of a track.

```
TimeValue GetTrackDuration (
   Track theTrack
);
```

Parameters

theTrack

The track for this operation. Your application obtains this track identifier from such functions as `NewMovieTrack` (page 112) and `GetMovieTrack` (page 85).

Return Value
The duration of the specified track, expressed in the time scale of the movie that contains the track.

Discussion
The duration corresponds to the ending time of the track in the movie's time coordinate system (remember that all tracks start at movie time 0).

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
BurntTextSampleCode
MakeEffectMovie
makeeffectslideshow
makeeffectslideshow.win
qteffects.win

Declared In
`Movies.h`

GetTrackEditRate

Returns the rate of the track edit of a specified track at an indicated time.

```
Fixed GetTrackEditRate (
   Track theTrack,
   TimeValue atTime
);
```

Parameters

theTrack

The track identifier for which the rate of a track edit (at the time given in the `atTime` parameter) is to be determined.

atTime

Indicates a time value at which the rate of a track edit (of a track identified in the parameter `theTrack`) is to be determined.

Return Value
The rate of the track edit of the specified track at the specified time.

Discussion
This function is useful if you are stepping through track edits directly in your application or if you are a client of QuickTime's base media handler.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
addflashactions.win
addhtactions.win
MovieVideoChart
qtwiredactions
TimeCode Media Handlers

Declared In
`Movies.h`

GetTrackEditRate64

Returns the rate of the track edit of a specified track at an indicated time.

```
Fixed GetTrackEditRate64 (
   Track theTrack,
   TimeValue64 atTime
);
```

Parameters

theTrack

A track identifier, which your application obtains from such functions as `NewMovieTrack` (page 112) and `GetMovieTrack` (page 85).

atTime

A 64-bit time value that indicates the time at which the rate of a track edit (of a track identified in the parameter `theTrack`) is to be determined.

Return Value
The rate of the track edit of the specified track at the specified time.

Discussion
This function is useful if you are stepping through track edits directly in your application or if you are a client of QuickTime's base media handler.

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

Declared In
`Movies.h`

GetTrackEnabled

Determines whether a track is currently enabled.

```
Boolean GetTrackEnabled (
   Track theTrack
);
```

Parameters

theTrack

The track for this operation. Your application obtains this track identifier from such functions as `NewMovieTrack` and `GetMovieTrack` (page 85).

Return Value
TRUE if the specified track is currently enabled, FALSE otherwise.

Discussion
The Movie Toolbox services only enabled tracks.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
QTKitTimeCode
qttext
qttext.win
qttimecode
vrscript

Declared In
`Movies.h`

GetTrackID

Determines a track's unique track ID value.

```
long GetTrackID (
   Track theTrack
);
```

Parameters

theTrack

The track for this operation. Your application obtains this track identifier from such functions as `NewMovieTrack` (page 112) and `GetMovieTrack` (page 85).

Return Value
The specified track's unique track ID value.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
QT Internals

vrmakepano
VRMakePano Library
vrmakepano.win

Declared In
Movies.h

GetTrackLayer

Retrieves a track's layer.

```
short GetTrackLayer (
   Track theTrack
);
```

Parameters

theTrack

The track for this operation. Your application obtains this track identifier from such functions as NewMovieTrack (page 112) and GetMovieTrack (page 85).

Return Value
The specified track's layer number. Layers with lower numbers appear in front of layers with higher numbers.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
BurntTextSampleCode
qtaddeffectseg
qtaddeffectseg.win
qteffects
qteffects.win

Declared In
Movies.h

GetTrackMatrix

Retrieves a track's transformation matrix.

```
void GetTrackMatrix (
   Track theTrack,
   MatrixRecord *matrix
);
```

Parameters

theTrack

The track for this operation. Your application obtains this track identifier from such functions as NewMovieTrack and GetMovieTrack (page 85).

matrix

A pointer to a `MatrixRecord` structure. The `GetTrackMatrix` function returns the track's matrix into the structure referred to by this parameter.

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

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
BurntTextSampleCode
QTKitTimeCode
qttext.win
qttimecode.win
TimeCode Media Handlers

Declared In
`Movies.h`

GetTrackMedia

Determines the media that contains a track's sample data.

```
Media GetTrackMedia (
   Track theTrack
);
```

Parameters

theTrack

The track for this operation. Your application obtains this track identifier from such functions as `NewMovieTrack` (page 112) and `GetMovieTrack` (page 85).

Return Value
The media identifier for the media that contains the track's sample data. If the function could not locate the media, it sets this returned value to `NIL`.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
MakeEffectMovie
MovieVideoChart
qtspritesplus.win
qttext
qttext.win

Declared In
Movies.h

GetTrackModificationTime

Returns a track's modification date and time.

```
unsigned long GetTrackModificationTime (
   Track theTrack
);
```

Parameters

theTrack

The track for this operation. Your application obtains this track identifier from such functions as NewMovieTrack (page 112) and GetMovieTrack (page 85).

Return Value
The specified track's modification date and time.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
Movies.h

GetTrackMovie

Determines the movie that contains a specified track.

```
Movie GetTrackMovie (
   Track theTrack
);
```

Parameters

theTrack

The track for this operation. Your application obtains this track identifier from such functions as NewMovieTrack and GetMovieTrack (page 85).

Return Value
The identifier of the movie that contains the track.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
MakeEffectMovie
qtmovietrack
qtmovietrack.win

qttext
qttext.win

Declared In
Movies.h

GetTrackOffset

Determines the time difference between the start of a track and the start of the movie that contains the track.

```
TimeValue GetTrackOffset (
   Track theTrack
);
```

Parameters

theTrack

The track for this operation. Your application obtains this track identifier from such functions as NewMovieTrack (page 112) and GetMovieTrack (page 85).

Return Value
The time difference between the start of the specified track and the start of the movie that contains the track.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
addflashactions.win
addhtactions.win
BurntTextSampleCode
qtwiredactions
qtwiredactions.win

Declared In
Movies.h

GetTrackReference

Retrieves the track identifier contained in an existing track reference.

```
Track GetTrackReference (
   Track theTrack,
   OSType refType,
   long index
);
```

Parameters

theTrack

Identifies the track for this operation. Your application obtains this track identifier from such functions as NewMovieTrack (page 112) and GetMovieTrack (page 85).

refType
The type of reference; see `Data References`.

index
The index value of the reference found. You obtain this index value when you create the track reference.

Return Value
The track identifier for the specified track. If the toolbox cannot locate the track reference corresponding to your specifications, it returns a value of `NIL`.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
MakeEffectMovie
QTKitTimeCode
qttext
qttext.win
vrmovies.win

Declared In
`Movies.h`

GetTrackReferenceCount

Determines how many track references of a given type exist for a track.

```
long GetTrackReferenceCount (
   Track theTrack,
   OSType refType
);
```

Parameters

theTrack
Identifies the track for this operation. Your application obtains this track identifier from such functions as `NewMovieTrack` (page 112) and `GetMovieTrack` (page 85).

refType
The type of reference; see `Data References`. The toolbox determines the number of track references of this type.

Return Value
A long integer that specifies the number of track references of the specified type in the track. If there are no references of the type you have specified, the function returns a value of 0.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
MakeEffectMovie

QTKitTimeCode
qttext
qttext.win
vrmovies.win

Declared In
`Movies.h`

GetTrackSoundLocalizationSettings

Returns a handle to a copy of the current 3D sound settings for a specified track.

```
OSErr GetTrackSoundLocalizationSettings (
   Track theTrack,
   Handle *settings
);
```

Parameters

theTrack

Identifies the track for this operation. Your application obtains this track identifier from such functions as `NewMovieTrack` (page 112) and `GetMovieTrack` (page 85).

settings

A handle to a copy of the current 3D sound settings for a specified track, in the format of an `SSpLocalizationData` record. If there are no 3D sound settings, the returned handle is set to `NIL`.

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

Special Considerations

The caller of this function is responsible for disposing of the returned handle.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`Movies.h`

GetTrackUsage

Determines whether a track is used in a movie, its preview, its poster, or a combination of these.

```
long GetTrackUsage (
   Track theTrack
);
```

Parameters

theTrack

The track for this operation. Your application obtains this track identifier from such functions as `NewMovieTrack` (page 112) and `GetMovieTrack` (page 85).

Return Value

`Track` usage flags (see below). These flags may be combined.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

QT Internals

qtinfo

qtinfo.win

Declared In

`Movies.h`

GetTrackUserData

Obtains access to a track's user data list.

```
UserData GetTrackUserData (
   Track theTrack
);
```

Parameters

theTrack

The track for this operation. Your application obtains this track identifier from such functions as `NewMovieTrack` (page 112) and `GetMovieTrack` (page 85).

Return Value

A reference to the specified track's user data. If the function could not locate the track's user data, it sets this returned value to `NIL`.

Discussion

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

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

MakeEffectMovie

Movie From DataRef
qteffects.win
vrbackbuffer
vrmovies.win

Declared In
Movies.h

GetTrackVolume

Returns a track's current volume setting.

```
short GetTrackVolume (
   Track theTrack
);
```

Parameters

theTrack

The track for this operation. Your application obtains this track identifier from such functions as NewMovieTrack (page 112) and GetMovieTrack (page 85).

Return Value
The specified track's current volume setting. The values returned in the high and low words range from 0x0000 (silence) to 0x0100 (full volume). You can use constants (see below) to test for full volume and no volume.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
BurntTextSampleCode
MovieGWorlds
QT Internals
vrmakepano
vrmakepano.win

Declared In
Movies.h

InsertEmptyMovieSegment

Adds an empty segment to a movie.

```
OSErr InsertEmptyMovieSegment (
   Movie dstMovie,
   TimeValue dstIn,
   TimeValue dstDuration
);
```

Parameters

dstMovie

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

dstIn

A time value that specifies where the segment is to be inserted. This time value must be expressed in the movie's time scale.

dstDuration

A time value that specifies the duration of the segment to be added. This time value must be expressed in the movie's time scale.

Return Value

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

Discussion

You specify the starting time and duration of the empty segment to be added. These times must be expressed in the movie's time scale. You cannot add empty space to the end of a movie. If you want to insert a segment beyond the end of a movie, use `InsertMovieSegment` (page 106).

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Movies.h`

InsertEmptyTrackSegment

Adds an empty segment to a track.

```
OSErr InsertEmptyTrackSegment (
   Track dstTrack,
   TimeValue dstIn,
   TimeValue dstDuration
);
```

Parameters

dstTrack

The track for this operation. Your application obtains this track identifier from such functions as `NewMovieTrack` (page 112) and `GetMovieTrack` (page 85).

dstIn

A time value specifying where the segment is to be inserted. This time value must be expressed in the time scale of the movie that contains the destination track.

dstDuration

A time value that specifies the duration of the segment to be added. This time value must be expressed in the time scale of the movie that contains the destination track.

Return Value

See `Error Codes`. If you try to add an empty segment beyond the end of a track, this function does not add the empty segment and returns a result code of `invalidTime`. Returns `noErr` if there is no error.

Discussion

You specify the starting time and duration of the empty segment to be added. These times must be expressed in the movie's time scale. This function then inserts the appropriate amount of empty time into the track. The exact meaning of the term empty time depends upon the type of track. For example, empty time in a sound track is silence. Note that you cannot add empty space to the end of a movie or to the end of a track.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Movies.h`

InsertMediaIntoTrack

Inserts a reference to a media segment into a track.

```
OSErr InsertMediaIntoTrack (
   Track theTrack,
   TimeValue trackStart,
   TimeValue mediaTime,
   TimeValue mediaDuration,
   Fixed mediaRate
);
```

Parameters

theTrack

The track for this operation. Your application obtains this track identifier from such functions as `NewMovieTrack` (page 112) or `GetMovieTrack` (page 85).

trackStart

A time value specifying where the segment is to be inserted. This time value must be expressed in the movie's time scale. If you set this parameter to -1, the media data is added to the end of the track.

mediaTime

A time value specifying the starting point of the segment in the media. This time value must be expressed in the media's time scale.

mediaDuration

A time value specifying the duration of the media's segment. This time value must be expressed in the media's time scale.

mediaRate

The media's rate. A value of 1.0 indicates the media's natural playback rate. This value should be positive and not 0.

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

Discussion
You specify the segment in the media by providing a starting time and duration. You specify the point in the destination track by providing a time in the track. `InsertMediaIntoTrack` then inserts the media segment into the track at the specified location. The Movie Toolbox determines the duration of the segment in the track based on the media rate and duration information you provide.

Use this function after you have added samples to a media. If you play the track before you call this function, the track does not contain the new media data.

Here's an example of using this function to add atom containers to a track:

```
//InsertMediaIntoTrack coding example
long descSize;
QTVRSampleDescriptionHandle qtvrSampleDesc;

// Create a QTVR sample description handle
descSize =sizeof(QTVRSampleDescription) + GetHandleSize((Handle) vrWorld)
                            - sizeof(UInt32);
qtvrSampleDesc =(QTVRSampleDescriptionHandle) NewHandleClear (descSize);
(*qtvrSampleDesc)->
size =descSize;
(*qtvrSampleDesc)->
type =kQTVRQTVRType;

// Copy the VR world atom container data into the QTVR sample description
BlockMove (*((Handle) vrWorld), &((*qtvrSampleDesc)->
data),
                            GetHandleSize((Handle) vrWorld));
// Now add it to the QTVR track's media
err =BeginMediaEdits (qtvrMedia);
err =AddMediaSample (qtvrMedia, (Handle) nodeInfo, 0,
    GetHandleSize((Handle) nodeInfo), duration,
    (SampleDescriptionHandle) qtvrSampleDesc, 1, 0, &sampleTime);
err =EndMediaEdits (qtvrMedia);
InsertMediaIntoTrack (qtvrTrack, trackTime, sampleTime, duration, 1L<<16);
```

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
qteffects
qteffects.win
vrmakepano
VRMakePano Library
vrmakepano.win

Declared In
`Movies.h`

InsertMovieSegment

Copies part of one movie to another.

```
OSErr InsertMovieSegment (
   Movie srcMovie,
   Movie dstMovie,
   TimeValue srcIn,
   TimeValue srcDuration,
   TimeValue dstIn
);
```

Parameters

srcMovie

The source movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie`, `NewMovieFromFile`, and `NewMovieFromHandle`. This function obtains the movie segment from the source movie specified in this parameter.

dstMovie

The destination movie for this operation. The function places a copy of the segment, which it obtained from the source movie, into this destination movie.

srcIn

The start of the segment in the source movie. This time value must be expressed in the source movie's time scale.

srcDuration

The duration of the segment in the source movie. This time value must be expressed in the source movie's time scale.

dstIn

A time value specifying where the segment is to be inserted. This time value must be expressed in the destination movie's time scale.

Return Value

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

Discussion

If you are not copying data from one location in a movie to a different point in the same movie, this function may create new tracks, as appropriate. Before adding a track to the destination movie, the Movie Toolbox looks in the destination movie for tracks that have the same characteristics as the tracks in the source movie. The toolbox considers several characteristics when searching for an appropriate track, including track spatial dimensions, track matrix, track clipping region, track matte, alternate group affiliation, media time scale, media type, media language, and data reference (that is, referring to the same file). If the Movie Toolbox cannot find an appropriate track in the destination movie, it creates a new track with the proper characteristics.

Special Considerations

If you have assigned a progress function to the destination movie, the Movie Toolbox calls that progress function during long copy operations. Some Movie Toolbox functions can take a long time to execute. For example, if you call `FlattenMovie` and specify a large movie, the Movie Toolbox must read and write all the sample data for the movie. During such operations you may wish to display some kind of progress indicator to the user.

Version Notes

Introduced in QuickTime 3 or earlier.

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

Related Sample Code
bMoviePalette
bMoviePaletteCocoa
qtstreamsplicer
qtstreamsplicer.win

Declared In
`Movies.h`

InsertTrackSegment

Copies data into a track.

```
OSErr InsertTrackSegment (
   Track srcTrack,
   Track dstTrack,
   TimeValue srcIn,
   TimeValue srcDuration,
   TimeValue dstIn
);
```

Parameters

srcTrack
The source track for this operation. Your application obtains this track identifier from such functions as `NewMovieTrack` (page 112) and `GetMovieTrack` (page 85).

dstTrack
The destination track for this operation. This function places a copy of the segment, which is obtained from the source track, into this destination track. The media for the destination track must be opened for writing by calling `BeginMediaEdits` (page 33) in order for the data to be copied. If the media is not opened for writing, the segment will be copied by reference. At the end of the editing session, your application must call `EndMediaEdits` (page 51) if it has called `BeginMediaEdits`.

srcIn
The start of the segment in the source track. This time value must be expressed in the time scale of the movie that contains the source track.

srcDuration
The duration of the segment in the source track. This time value must be expressed in the time scale of the movie that contains the source track.

dstIn
A time value specifying where the segment is to be inserted. This time value must be expressed in the time scale of the movie that contains the destination track.

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

Discussion
If you are copying data between tracks, make sure that the two tracks are of the same type. For example, you cannot copy a segment from a sound track into a video track. If you have assigned a progress function to the movie that contains the destination track, the Movie Toolbox calls that progress function during long copy operations.

Special Considerations

If you copy a segment without calling `BeginMediaEdits` on the destination track's media, the data can be copied later by flattening the movie.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
qtaddeffectseg
qtaddeffectseg.win
qteffects
qteffects.win
qtstreamsplicer.win

Declared In
`Movies.h`

IsScrapMovie

Checks the system scrap to find out if it can translate any of the data into a movie.

```
Component IsScrapMovie (
   Track targetTrack
);
```

Parameters

`targetTrack`
The location of the potential target movie track for the data on the system scrap.

Return Value
If `IsScrapMovie` finds an appropriate type, it returns a movie import component that can translate the scrap. Otherwise, it returns 0.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`Movies.h`

MediaContainsDisplayOffsets

Tests whether a media contains display offsets.

```
Boolean MediaContainsDisplayOffsets (
   Media theMedia
);
```

Parameters

theMedia

The media for this operation. You obtain this media identifier from such functions as `NewTrackMedia` (page 114) and `GetTrackMedia` (page 96).

Return Value

TRUE if the media is valid and contains at least one sample with a nonzero display offset; FALSE otherwise.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`Movies.h`

MediaDecodeTimeToSampleNum

Finds the sample for a specified decode time.

```
void MediaDecodeTimeToSampleNum (
   Media theMedia,
   TimeValue64 decodeTime,
   SInt64 *sampleNum,
   TimeValue64 *sampleDecodeTime,
   TimeValue64 *sampleDecodeDuration
);
```

Parameters

theMedia

The media for this operation. You obtain this media identifier from such functions as `NewTrackMedia` (page 114) and `GetTrackMedia` (page 96).

decodeTime

A 64-bit time value that represents the decode time for which you are retrieving sample information. You must specify this value in the media's time scale.

sampleNum

A pointer to a variable that is to receive the sample number. The function returns the sample number that identifies the sample that contains data for the specified decode time, or 0 if it is not found.

sampleDecodeTime

A pointer to a time value. The function updates this time value to indicate the decode time of the sample specified by the `logicalSampleNum` parameter. This time value is expressed in the media's time scale. Set this parameter to NULL if you do not want this information.

sampleDecodeDuration

A pointer to a time value. The function updates this time value to indicate the decode duration of the sample specified by the `logicalSampleNum` parameter. This time value is expressed in the media's time scale. Set this parameter to NULL if you do not want this information.

Discussion
You can access this function's error returns through `GetMoviesError` and `GetMoviesStickyError`. It returns `paramErr` if there is a bad parameter value, `invalidTime` if `sampleDecodeTime` is out of the decode time range, or `noErr` if there is no error.

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

Related Sample Code
MovieVideoChart

Declared In
`Movies.h`

MediaDisplayTimeToSampleNum

Finds the sample number for a specified display time.

```
void MediaDisplayTimeToSampleNum (
   Media theMedia,
   TimeValue64 displayTime,
   SInt64 *sampleNum,
   TimeValue64 *sampleDisplayTime,
   TimeValue64 *sampleDisplayDuration
);
```

Parameters

theMedia

The media for this operation. You obtain this media identifier from such functions as `NewTrackMedia` (page 114) and `GetTrackMedia` (page 96).

displayTime

A 64-bit time value that represents the display time for which you are retrieving sample information. You must specify this value in the media's time scale.

sampleNum

A pointer to a long integer that is to receive the sample number. The function returns the sample number that identifies the sample for the specified display time, or 0 if it is not found.

sampleDisplayTime

A pointer to a time value. The function updates this time value to indicate the display time of the sample specified by the `logicalSampleNum` parameter. This time value is expressed in the media's time scale. Set this parameter to NULL if you do not want this information.

sampleDisplayDuration

A pointer to a time value. The function updates this time value to indicate the display duration of the sample specified by the `logicalSampleNum` parameter. This time value is expressed in the media's time scale. Set this parameter to NULL if you do not want this information.

Discussion
You can access this function's error returns through `GetMoviesError` and `GetMoviesStickyError`. It returns `paramErr` if there is a bad parameter value, `invalidTime` if `sampleDisplayTime` is out of the display time range, or `noErr` if there is no error.

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

Related Sample Code
MovieVideoChart

Declared In
`Movies.h`

MediaTimeToSampleNum

Lets you find the sample that contains the data for a specified time.

```
void MediaTimeToSampleNum (
   Media theMedia,
   TimeValue time,
   long *sampleNum,
   TimeValue *sampleTime,
   TimeValue *sampleDuration
);
```

Parameters

theMedia

The media for this operation. You obtain this media identifier from such functions as `NewTrackMedia` (page 114) and `GetTrackMedia` (page 96).

time

The time for which you are retrieving sample information. You must specify this value in the media's time scale.

sampleNum

A pointer to a long integer that is to receive the sample number. The Movie Toolbox returns the sample number that identifies the sample that contains data for the time specified by the `time` parameter.

sampleTime

A pointer to a time value. The `MediaTimeToSampleNum` function updates this time value to indicate the starting time of the sample that contains data for the time specified by the `time` parameter. This time value is expressed in the media's time scale. Set this parameter to `NIL` if you don't want this information.

sampleDuration

A pointer to a time value. The Movie Toolbox returns the duration of the sample that contains data for the time specified by the `time` parameter. This time value is expressed in the media's time scale. Set this parameter to `NIL` if you don't want this information.

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

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
qttext
qttext.win

Declared In
`Movies.h`

NewMovieEditState

Creates an edit state.

```
MovieEditState NewMovieEditState (
   Movie theMovie
);
```

Parameters

theMovie

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

Return Value
A pointer to a `MovieEditStateRecord` structure. The edit state contains all the information describing a movie's content, including the current selection, the movie's tracks, and the media data associated with those tracks.

Special Considerations

You must dispose of a movie's `MovieEditStateRecord` structures, using `DisposeMovieEditState` (page 48), before you dispose of the movie itself.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`Movies.h`

NewMovieTrack

Creates a new movie track, without a media.

```
Track NewMovieTrack (
   Movie theMovie,
   Fixed width,
   Fixed height,
   short trackVolume
);
```

Parameters

theMovie

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

width

A fixed number denoting the display width of the track, in pixels.

height

A fixed number denoting the display height of the track, in pixels. Together, the `height` and width parameters define the track's display rectangle. The upper-left corner of this rectangle lies at (0,0) in the movie's rectangle. The height and width parameters therefore establish the lower-right corner of the track's display rectangle. If you are creating a track that is not displayed, such as a sound track, set the `height` and width parameters to 0.

trackVolume

The volume setting of the track as a 16-bit, fixed-point number. The high-order 8 bits specify the integer portion; the low-order 8 bits specify the fractional part. Volume values range from -1.0 to 1.0. Negative values play no sound but preserve the absolute value of the volume setting. Set this parameter to `kFullVolume` to play the track at its full, natural volume. Set this parameter to `kNoVolume` to set the volume to 0. See these constants:

Return Value
The identifier of the new track.

Discussion
Immediately after creating a new track, you should call `NewTrackMedia` (page 114) to create a media for the track; a track without a media is of no use. The following code sample creates a new sprite track and media, then calls `BeginMediaEdits` (page 33) to prepare to add samples to the media:

```
// NewMovieTrack coding example
// See "Discovering QuickTime," page 349
#define kSpriteMediaTimeScale           600
track =NewMovieTrack(movie, ((long)lTrackWidth << 16),
                            ((long)lTrackHeight << 16), 0);
media =NewTrackMedia(track, SpriteMediaType,
                            kSpriteMediaTimeScale, NIL, 0);
FailOSErr(BeginMediaEdits(media));
```

Special Considerations

When you add a track to a movie, the Movie Toolbox automatically adjusts the display `Rect` structure of the movie. You may want to detect these changes by calling `GetMovieBox` so that you can adjust the size of the movie's display window.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
qteffects
qteffects.win
vrmakepano
VRMakePano Library
vrmakepano.win

Declared In
`Movies.h`

NewTrackEditState

Creates a new edit state for a given track.

```
TrackEditState NewTrackEditState (
   Track theTrack
);
```

Parameters

theTrack

The track for this operation. Your application obtains this track identifier from such functions as `NewMovieTrack` (page 112) and `GetMovieTrack` (page 85).

Return Value

The track's edit state identifier. If the edit state could not be created, the returned identifier is set to `NIL`. You must dispose of a movie's track edit states, using Use `DisposeTrackEditState` (page 50), before disposing of the track or of the movie that contains the track.

Discussion

Use the returned identifier with other Movie Toolbox edit state functions, such as `UseTrackEditState` (page 144). The edit state contains all the information describing a track's content, including the identity of the media data associated with the track and all the track's edit lists.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Movies.h`

NewTrackMedia

Creates a media for a new track.

```
Media NewTrackMedia (
   Track theTrack,
   OSType mediaType,
   TimeScale timeScale,
   Handle dataRef,
   OSType dataRefType
);
```

Parameters

theTrack

The track for this operation. Your application obtains this track identifier from such functions as `NewMovieTrack` (page 112).

mediaType

The type of media to create; see `Media Identifiers`. The Movie Toolbox uses this value to find the correct media handler for the new media. If the Movie Toolbox cannot locate an appropriate media handler, it returns an error.

timeScale

Defines the media's time coordinate system.

dataRef

The data reference. This parameter contains a handle to the information that identifies the file that contains this media's data. The type of information stored in that handle depends upon the value of the `dataRefType` parameter. If you are creating a new media that refers to existing media data, you can use the `GetMediaDataRef` function to obtain information about the existing data reference. You can then supply information about that reference to this function. Set this parameter to `NIL` to use the file that is associated with the movie or if the movie does not have a movie file. For example, if you have created the movie using `CreateMovieFile` or `NewMovieFromFile`, the Movie Toolbox assumes that the movie's data resides in the file specified at that time. If you have created the movie using the `NewMovieFromScrap` or `NewMovie` functions, the movie does not have a movie file.

dataRefType

The type of data reference; see `Data References`. If the data reference is an alias, you must set this parameter to `rAliasType`. See *Inside Macintosh: Files* for more information about aliases and the Alias Manager.

Return Value
A media identifier, referring to the actual data samples used by the track. If the function cannot create a new media, it sets the returned value to `NIL`.

Discussion
The following code sample creates a new sprite track and media, then calls `BeginMediaEdits` (page 33) to prepare to add samples to the media:

```
// NewTrackMedia coding example
// See "Discovering QuickTime," page 349
#define kSpriteMediaTimeScale           600
track =NewMovieTrack(movie, ((long)lTrackWidth << 16),
                            ((long)lTrackHeight << 16), 0);
media =NewTrackMedia(track, SpriteMediaType,
                            kSpriteMediaTimeScale, NIL, 0);
FailOSErr(BeginMediaEdits(media));
```

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
qteffects
qteffects.win
vrmakepano
VRMakePano Library
vrmakepano.win

Declared In
`Movies.h`

OpenADataHandler

Opens a data handler component.

```
OSErr OpenADataHandler (
   Handle dataRef,
   OSType dataHandlerSubType,
   Handle anchorDataRef,
   OSType anchorDataRefType,
   TimeBase tb,
   long flags,
   ComponentInstance *dh
);
```

Parameters

dataRef

A handle to a data reference. The type of information stored in the handle depends upon the `data` reference type specified by the `dataHandlerSubType` parameter.

dataHandlerSubType

Identifies both the type of data reference and, by implication, the `component` subtype value assigned to the data handler components that operate on data references of that type.

anchorDataRef

A handle to the anchor data reference.

anchorDataRefType

The type of the anchor data reference.

tb

The time base for the data handler. Your application obtains this time base identifier from `NewTimeBase`.

flags

Flags (see below) that indicate the way in which you intend to use the data handler component. Not all data handlers necessarily support all services; for example, some data handler components may not support streaming writes. Set the appropriate flags to 1. See these constants:

```
kDataHCanRead
kDataHCanWrite
kDataHCanStreamingWrite
```

dh

A pointer to a field to receive the `ComponentInstance` value of the newly-opened data handler component.

Return Value

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

Version Notes

Introduced in QuickTime 4.1.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

ElectricImageComponent

ElectricImageComponent.win

Declared In

`Movies.h`

PasteHandleIntoMovie

Takes the contents of a specified handle, together with its type, and pastes it into a specified movie.

```
OSErr PasteHandleIntoMovie (
   Handle h,
   OSType handleType,
   Movie theMovie,
   long flags,
   ComponentInstance userComp
);
```

Parameters

h

The handle to be pasted into the movie indicated by the `theMovie` parameter.

handleType

The data type of the handle specified in the `h` parameter. If the handle is set to 0, the function searches the scrap for a field of the type `handleType`. If both the `h` parameter and the `handleType` parameter are `NIL`, the function uses the first available data from the scrap.

theMovie

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

flags

A flag (see below) that can further refine conditions of the paste operation. See these constants:

`pasteInParallel`

userComp

The component or an instance of the component that is to perform the conversion of the data into a QuickTime movie. If you want a particular movie import component to perform the conversion, you may pass the component or an instance of that component. Otherwise, set this parameter to 0 to allow the Movie Toolbox to determine the appropriate component. If you pass in a component instance, this function uses it. This allows you to communicate directly with the component before using this function to establish any conversion parameters. If you pass in a component ID, an instance is created and closed within this function.

Return Value

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

Discussion

If you are just pasting in data from the scrap, it is best to allow this function to retrieve the data from the scrap, rather than doing it yourself. In this way, the function is able to obtain supplemental data from the scrap, if necessary (for example, `'styl'` resources for `'TEXT'`). This function can paste into the current selection in two different ways. If the selection is empty (for example, duration =0), it adds the data with the appropriate duration. If the selection is not empty, the data is added and then scaled to fit into the duration of the selection. The current selection is deleted, unless you set the `pasteInParallel` flag.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In
Movies.h

PasteMovieSelection

Places the tracks from one movie into another movie.

```
void PasteMovieSelection (
   Movie theMovie,
   Movie src
);
```

Parameters

theMovie

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

src

The source movie for this operation. PasteMovieSelection places the tracks from this movie in the destination movie.

Return Value
You can access error returns from this function through GetMoviesError and GetMoviesStickyError. See Error Codes.

Discussion
Whenever possible, the Movie Toolbox uses existing tracks to store the data to be pasted. Before adding a track to the destination movie, the Toolbox looks in the destination movie for tracks that have the same characteristics as the tracks in the source movie. It considers several characteristics when searching for an appropriate track, including track spatial dimensions, track matrix, track clipping region, track matte, alternate group affiliation, media time scale, media type, media language, and data reference (that is, the two tracks must refer to the same file). If the Movie Toolbox cannot find an appropriate track in the destination movie, it creates a track with the proper characteristics. It removes any empty tracks from the destination movie after the paste operation.

Special Considerations

If you have assigned a progress function to the destination movie, the Movie Toolbox calls that progress function during long paste operations.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
Movies.h

PtInMovie

Determines whether a specified point lies in the region defined by a movie's final display boundary region after it has been clipped by the movie's display clipping region.

```
Boolean PtInMovie (
   Movie theMovie,
   Point pt
);
```

Parameters

theMovie

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

pt

The point to be checked. This point must be expressed in the movie's local display coordinate system.

Return Value
Returns TRUE if the point is in the movie.

Discussion
This function is accurate at the current movie time.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`Movies.h`

PtInTrack

Determines whether a specified point lies in the region defined by a track's display boundary region after it has been clipped by the movie's final display clipping region.

```
Boolean PtInTrack (
   Track theTrack,
   Point pt
);
```

Parameters

theTrack

The track for this operation. Your application obtains this track identifier from such functions as `NewMovieTrack` (page 112) and `GetMovieTrack` (page 85).

pt

The point to be checked. This point must be expressed in the local display coordinate system of the movie that contains the track.

Return Value
Returns TRUE if the point lies in the track's display space.

Discussion
This function is accurate at the current movie time.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`Movies.h`

PutMovieIntoTypedHandle

Takes a movie, or a single track from within that movie, and converts it into a handle of a specified type.

```
OSErr PutMovieIntoTypedHandle (
   Movie theMovie,
   Track targetTrack,
   OSType handleType,
   Handle publicMovie,
   TimeValue start,
   TimeValue dur,
   long flags,
   ComponentInstance userComp
);
```

Parameters

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

targetTrack
: The track to convert.

handleType
: The type of the new data.

publicMovie
: The actual handle in which to place the new data.

start
: The start time of the segment of the movie or track to be converted.

dur
: The duration of the segment of the movie or track to be converted.

flags
: Condition of the conversion. Set this parameter to 0.

userComp
: Indicates a component or component instance of the movie export component you want to perform the conversion. Otherwise, set this parameter to 0 for the Movie Toolbox to choose the appropriate component. If you pass in a component instance, this function will use it. This allows you to communicate directly with the component before using this function to establish any conversion parameters. If you pass in a component ID, an instance is created and closed within this function.

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

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
CompressMovies
DigitizerShell
DragAndDrop Shell
MovieGWorlds
soundsnippets

Declared In
`Movies.h`

QTGetMIMETypeInfo

Retrieves information about a particular MIME type.

```
OSErr QTGetMIMETypeInfo (
   const char *mimeStringStart,
   short mimeStringLength,
   OSType infoSelector,
   void *infoDataPtr,
   long *infoDataSize
);
```

Parameters

mimeStringStart
A pointer to the first character of a string holding the MIME type.

mimeStringLength
The number of characters in the MIME type string. Pascal, C, and nondelimited string buffers can be passed equally well.

infoSelector
A constant (see below) that indicates the type of information being requested. See these constants:
`kQTGetMIMETypeInfoIsQuickTimeMovieType`
`kQTGetMIMETypeInfoIsUnhelpfulType`

infoDataPtr
A pointer to a value to be updated. For current selectors this value is Boolean.

infoDataSize
On input, a pointer to the size of the data being expected; on output, a pointer to the size of the data being retrieved. In all current cases these will be the same size.

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

Version Notes
Introduced in QuickTime 5.

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

Declared In
Movies.h

SampleNumToMediaDecodeTime

Finds the decode time for a specified sample.

```
void SampleNumToMediaDecodeTime (
   Media theMedia,
   SInt64 logicalSampleNum,
   TimeValue64 *sampleDecodeTime,
   TimeValue64 *sampleDecodeDuration
);
```

Parameters

theMedia

The media for this operation. You obtain this media identifier from such functions as NewTrackMedia (page 114) and GetTrackMedia (page 96).

logicalSampleNum

A 64-bit signed integer that contains the sample number.

sampleDecodeTime

A pointer to a time value. The function updates this time value to indicate the decode time of the sample specified by the logicalSampleNum parameter. This time value is expressed in the media's time scale. Set this parameter to NULL if you do not want this information.

sampleDecodeDuration

A pointer to a time value. The function updates this time value to indicate the decode duration of the sample specified by the logicalSampleNum parameter. This time value is expressed in the media's time scale. Set this parameter to NULL if you do not want this information.

Discussion
You can access this function's error returns through GetMoviesError and GetMoviesStickyError. It returns paramErr if there is a bad parameter value, or noErr if there is no error.

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

Related Sample Code
MovieVideoChart

Declared In
Movies.h

SampleNumToMediaDisplayTime

Finds the display time for a specified sample.

```
void SampleNumToMediaDisplayTime (
   Media theMedia,
   SInt64 logicalSampleNum,
   TimeValue64 *sampleDisplayTime,
   TimeValue64 *sampleDisplayDuration
);
```

Parameters

theMedia

The media for this operation. You obtain this media identifier from such functions as `NewTrackMedia` (page 114) and `GetTrackMedia` (page 96).

logicalSampleNum

A 64-bit signed integer that contains the sample number.

sampleDisplayTime

A pointer to a time value. The function updates this time value to indicate the display time of the sample specified by the `logicalSampleNum` parameter. This time value is expressed in the media's time scale. Set this parameter to NULL if you do not want this information.

sampleDisplayDuration

A pointer to a time value. The function updates this time value to indicate the display duration of the sample specified by the `logicalSampleNum` parameter. This time value is expressed in the media's time scale. Set this parameter to NULL if you do not want this information.

Discussion

You can access this function's error returns through `GetMoviesError` and `GetMoviesStickyError`. It returns `paramErr` if there is a bad parameter value, or `noErr` if there is no error.

Availability

Available in Mac OS X v10.3 and later.

Related Sample Code

MovieVideoChart

Declared In

`Movies.h`

SampleNumToMediaTime

Finds the time at which a specified sample plays.

```
void SampleNumToMediaTime (
   Media theMedia,
   long logicalSampleNum,
   TimeValue *sampleTime,
   TimeValue *sampleDuration
);
```

Parameters

theMedia

The media for this operation. You obtain this media identifier from such functions as `NewTrackMedia` (page 114) and `GetTrackMedia` (page 96).

logicalSampleNum

The sample number.

sampleTime

A pointer to a time value. The function updates this time value to indicate the starting time of the sample specified by the `logicalSampleNum` parameter. This time value is expressed in the media's time scale. Set this parameter to `NIL` if you don't want this information.

sampleDuration

A pointer to a time value. The Movie Toolbox returns the duration of the sample specified by the `logicalSampleNum` parameter. This time value is expressed in the media's time scale. Set this parameter to `NIL` if you don't want this information.

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

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`Movies.h`

ScaleMovieSegment

Changes the duration of a segment of a movie.

```
OSErr ScaleMovieSegment (
   Movie theMovie,
   TimeValue startTime,
   TimeValue oldDuration,
   TimeValue newDuration
);
```

Parameters

theMovie

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

startTime

The start of the segment. The `oldDuration` parameter specifies the segment's duration. This time value must be expressed in the movie's time scale.

oldDuration

The original duration of the segment in the source movie. This time value must be expressed in the movie's time scale.

newDuration

The new duration of the segment. This time value must be expressed in the movie's time scale. The function alters the segment to accommodate the new duration.

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

Discussion
The Movie Toolbox scales the segment to accommodate the new duration.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
SlideShowImporter
SlideShowImporter.win

Declared In
`Movies.h`

ScaleTrackSegment

Changes the duration of a segment of a track.

```
OSErr ScaleTrackSegment (
   Track theTrack,
   TimeValue startTime,
   TimeValue oldDuration,
   TimeValue newDuration
);
```

Parameters

theTrack
The track for this operation. Your application obtains this track identifier from such functions as `NewMovieTrack` (page 112) and `GetMovieTrack` (page 85).

startTime
The start of the segment. The `oldDuration` parameter specifies the segment's duration. This time value must be expressed in the time scale of the movie that contains the track.

oldDuration
The duration of the segment. This time value must be expressed in the time scale of the movie that contains the track.

newDuration
The new duration of the segment. This time value must be expressed in the time scale of the movie that contains the track. The function alters the segment to accommodate the new duration.

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

Discussion
This function does not cause the Movie Toolbox to add data to or remove data from the movie.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
makeeffectslideshow

makeeffectslideshow.win

Declared In
`Movies.h`

SelectMovieAlternates

Instructs the Movie Toolbox to select appropriate tracks immediately.

```
void SelectMovieAlternates (
   Movie theMovie
);
```

Parameters

theMovie

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

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

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`Movies.h`

SetAutoTrackAlternatesEnabled

Enables or disables automatic track selection by the Movie Toolbox.

```
void SetAutoTrackAlternatesEnabled (
   Movie theMovie,
   Boolean enable
);
```

Parameters

theMovie

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

enable

Controls automatic track selection. Set this parameter to TRUE to enable automatic track selection. Set this parameter to FALSE to disable automatic track selection.

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

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`Movies.h`

SetMediaDataHandler

Assigns a data handler to a media.

```
OSErr SetMediaDataHandler (
   Media theMedia,
   short index,
   DataHandlerComponent dataHandler
);
```

Parameters

theMedia

The media for this operation. Your application obtains this media identifier from such functions as `NewTrackMedia` (page 114) and `GetTrackMedia` (page 96). See `Media Identifiers`.

index

Identifies the data reference for this data handler. You provide the index value that corresponds to the data reference. You must set this parameter to 1.

dataHandler

The data handler for the media. This identifier is a component instance that specifies a connection to a data handler component, such as that returned by `GetMediaDataHandler` (page 55). If the data handler you specify cannot work with the data stored in the media, the function does not change the media's data handler.

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

Discussion
Your application should normally not call this function. The Movie Toolbox assigns a data handler to each media when you load a movie.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`Movies.h`

SetMediaDefaultDataRefIndex

Specifies which of a media's data references is to be accessed during an editing session.

```
OSErr SetMediaDefaultDataRefIndex (
   Media theMedia,
   short index
);
```

Parameters

theMedia

The media for this operation. Your application obtains this media identifier from such functions as `NewTrackMedia` (page 114) and `GetTrackMedia` (page 96). See `Media Identifiers`.

index

The data reference to access. Values of the `index` parameter range from 1 to the number of data references in the media. You can determine the number of data references by calling `GetMediaDataRefCount`. Once set, the default data reference index persists. Set this parameter to 0 to revert to the media's default data reference.

Return Value

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

Discussion

This function allows you to specify the index of the data reference to be edited. After calling this function, you can start editing that data reference by calling `BeginMediaEdits` (page 33).

Version Notes

Before QuickTime 2.0, the Movie Toolbox did not allow the creation of tracks that had data in several files. Therefore, there was no mechanism for controlling which data reference was affected by a media editing session.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Movies.h`

SetMediaHandler

Assigns a specific media handler to a track.

```
OSErr SetMediaHandler (
   Media theMedia,
   MediaHandlerComponent mH
);
```

Parameters

theMedia

The track for this operation. Your application obtains this track identifier from such functions as `NewMovieTrack` (page 112) and `GetMovieTrack` (page 85).

mH

A reference to a media handler component. You can obtain this reference from `GetMediaHandler` (page 61).

Return Value

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

Discussion
Your application should not need to call this function. The Movie Toolbox assigns a media handler to each track when you load a movie.

Special Considerations

The Movie Toolbox closes the track's previous media handler and then opens the new one. It is your responsibility to ensure that the media handler you specify can handle the data in the track.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`Movies.h`

SetMediaInputMap

Replaces the media's existing input map with a given input map.

```
OSErr SetMediaInputMap (
   Media theMedia,
   QTAtomContainer inputMap
);
```

Parameters

theMedia

The media for this operation. Your application obtains this media identifier from such functions as `NewTrackMedia` (page 114) and `GetTrackMedia` (page 96). See `Media Identifiers`.

inputMap

The media input map for this operation. If the input map is set to `NIL`, the media's input map is reset to an empty input map.

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

Discussion
Use this function to specify the media you want to set so you can modify or empty its input map. It makes a copy of the input map passed to it. The following sample code illustrates how to update an input map, using this function and `GetMediaInputMap` (page 63):

```
// SetMediaInputMap coding example
QTAtomContainer inputMap;
QTAtom inputAtom;
OSType inputType;
Media aVideoMedia =GetTrackMedia(aVideoTrack);
GetMediaInputMap (aVideoMedia, &inputMap);
QTInsertChild(inputMap, kParentAtomIsContainer, kTrackModifierInput,
        addedIndex, 0,0, nil, &inputAtom);
inputType =kTrackModifierTypeClip;
QTInsertChild (inputMap, inputAtom, kTrackModifierType, 1, 0,
        sizeof(inputType), &inputType, nil);
SetMediaInputMap(aVideoMedia, inputMap);
```

```
QTDisposeAtomContainer(inputMap);
```

Special Considerations

Use `QTNewAtomContainer` to create an empty input map.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
ChromaKeyMovie
qteffects
qteffects.win
qtshoweffect
qtshoweffect.win

Declared In
`Movies.h`

SetMediaLanguage

Sets a media's localized language or region code.

```
void SetMediaLanguage (
   Media theMedia,
   short language
);
```

Parameters

theMedia

The media for this operation. Your application obtains this media identifier from such functions as `NewTrackMedia` (page 114) and `GetTrackMedia` (page 96). See `Media Identifiers`.

language

The media's language or region code.

Return Value
You can access error returns from this function through `GetMoviesError` and `GetMoviesStickyError`. See `Error Codes`.

Discussion
You should call this function only when you are creating a new media.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`Movies.h`

SetMediaPreferredChunkSize

Specifies a maximum chunk size for a media.

```
OSErr SetMediaPreferredChunkSize (
   Media theMedia,
   long maxChunkSize
);
```

Parameters

theMedia

The media for this operation. Your application obtains this media identifier from such functions as `NewTrackMedia` (page 114) and `GetTrackMedia` (page 96). See `Media Identifiers`.

maxChunkSize

The maximum chunk size, in bytes.

Return Value

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

Discussion

The term "chunk" refers to the collection of sample data that is added to a movie when you call `AddMediaSample` (page 20). When QuickTime loads a movie for playback, it loads the data a chunk at a time. Consequently, both the size and number of chunks in a movie can affect playback performance. The toolbox tries to optimize playback performance by consolidating adjacent sample references into a single chunk, up to the limit you prescribe with this function.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Movies.h`

SetMediaQuality

Sets a media's quality level value.

```
void SetMediaQuality (
   Media theMedia,
   short quality
);
```

Parameters

theMedia

The media for this operation. Your application obtains this media identifier from such functions as `NewTrackMedia` (page 114) and `GetTrackMedia` (page 96). See `Media Identifiers`.

quality

The media's quality value. The quality value indicates the pixel depths at which the media can be played. This even applies to sound media. The low-order 6 bits of the quality value correspond to specific pixel depths. If a bit is set to 1, the media can be played at the corresponding depth. More than one of these bits may be set to 1. The Movie Toolbox uses this quality value to determine which track it selects to play on a given Macintosh computer. You should set this value only when you are creating a new media.

Return Value
You can access error returns from this function through `GetMoviesError` and `GetMoviesStickyError`. See `Error Codes`.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`Movies.h`

SetMediaSampleDescription

Changes the contents of a particular SampleDescription structure of a specified media.

```
OSErr SetMediaSampleDescription (
   Media theMedia,
   long index,
   SampleDescriptionHandle descH
);
```

Parameters

theMedia

The media for this operation. You obtain this media identifier from such functions as `NewTrackMedia` (page 114) and `GetTrackMedia` (page 96).

index

The index of the `SampleDescription` structure to be changed. This index corresponds to the `SampleDescription` structure itself, not the samples in the media. This long integer must be between 1 and the largest `SampleDescription` index.

descH

The handle to the `SampleDescription` structure. If there is no description for the specified index, the function returns this handle unchanged.

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

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
BurntTextSampleCode

Declared In
`Movies.h`

SetMediaShadowSync

Obsolete; no longer supported.

```
OSErr SetMediaShadowSync (
   Media theMedia,
   long frameDiffSampleNum,
   long syncSampleNum
);
```

Parameters

`theMedia`
The media in which the shadow sync is to be created.

`frameDiffSampleNum`
Specifies a frame difference sample. The sample number is obtained from `MediaTimeToSampleNum` (page 111).

`syncSampleNum`
Specifies a shadow sync sample. The sample number is obtained from `MediaTimeToSampleNum` (page 111).

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

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`Movies.h`

SetMediaTimeScale

Sets a media's time scale.

```
ComponentResult ADD_MEDIA_BASENAME() SetMediaTimeScale
```

Parameters

`theMedia`
The media for this operation. Your application obtains this media identifier from such functions as `NewTrackMedia` (page 114) and `GetTrackMedia` (page 96). See `Media Identifiers`.

`timeScale`
The media's new time scale.

Return Value
You can access error returns from this function through `GetMoviesError` and `GetMoviesStickyError`. See `Error Codes`.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`Movies.h`

SetTrackAlternate

Adds tracks to, or remove tracks from, alternate groups.

```
void SetTrackAlternate (
   Track theTrack,
   Track alternateT
);
```

Parameters

`theTrack`

The track and group for this operation. Your application obtains this track identifier from such functions as `NewMovieTrack` (page 112) and `GetMovieTrack` (page 85). `SetTrackAlternate` changes this track's group affiliation based on the value of the `alternateT` parameter.

`alternateT`

Controls whether the function adds the track to a group or removes it from a group. If this parameter contains a valid track identifier, the Movie Toolbox adds this track to the group that contains the track specified by the parameter `theTrack`. If the track identified by this parameter already belongs to a group, the Movie Toolbox combines the two groups into a single group.

Return Value
You can access error returns from this function through `GetMoviesError` and `GetMoviesStickyError`. See `Error Codes`.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`Movies.h`

SetTrackDimensions

Establishes a track's source rectangle.

```
void SetTrackDimensions (
   Track theTrack,
   Fixed width,
   Fixed height
);
```

Parameters

theTrack

The track for this operation. Your application obtains this track identifier from such functions as `NewMovieTrack` (page 112) and `GetMovieTrack` (page 85).

width

A fixed-point number that specifies the width, in pixels, of the track's rectangle. This value corresponds to the x coordinate of the lower-right corner of the track's rectangle.

height

A fixed-point number that specifies the height, in pixels, of the track's rectangle. This value corresponds to the y coordinate of the lower-right corner of the track's rectangle.

Return Value

You can access error returns from this function through `GetMoviesError` and `GetMoviesStickyError`. See `Error Codes`.

Discussion

If you change the dimensions of an existing track, the media data is scaled to fit into the new rectangle.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

QTKitTimeCode

qttimecode.win

SlideShowImporter

SlideShowImporter.win

TimeCode Media Handlers

Declared In

`Movies.h`

SetTrackEnabled

Enables or disables a track.

```
void SetTrackEnabled (
   Track theTrack,
   Boolean isEnabled
);
```

Parameters

theTrack

The track for this operation. Your application obtains this track identifier from such functions as `NewMovieTrack` (page 112) and `GetMovieTrack` (page 85).

isEnabled

Enables or disables the `track`. Set this parameter to TRUE to enable the `track`. Set this parameter to FALSE to disable the track.

Return Value
You can access error returns from this function through `GetMoviesError` and `GetMoviesStickyError`. See `Error Codes`.

Discussion
The Movie Toolbox services only enabled tracks.

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
vrmakepano
VRMakePano Library

Declared In
`Movies.h`

SetTrackLayer

Sets a track's layer.

```
void SetTrackLayer (
   Track theTrack,
   short layer
);
```

Parameters

theTrack

The track for this operation. Your application obtains this track identifier from such functions as `NewMovieTrack` (page 112) and `GetMovieTrack` (page 85).

layer

The track's layer number. Layers are numbered from -32,768 through 32,767; layers with lower numbers appear in front of layers with higher numbers. When you create a new track, the Movie Toolbox sets its track number to 0.

Return Value
You can access error returns from this function through `GetMoviesError` and `GetMoviesStickyError`. See `Error Codes`.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
BurntTextSampleCode
qtactiontargets
qtactiontargets.win
qtwiredspritesjr
qtwiredspritesjr.win

Declared In
`Movies.h`

SetTrackMatrix

Establishes a track's transformation matrix.

```
void SetTrackMatrix (
   Track theTrack,
   const MatrixRecord *matrix
);
```

Parameters

`theTrack`

The track for this operation. Your application obtains this track identifier from such functions as `NewMovieTrack` (page 112) and `GetMovieTrack` (page 85).

`matrix`

A pointer to a `MatrixRecord` structure that contains the track's new matrix. If you set this parameter to `NIL`, the Movie Toolbox uses the identity matrix.

Return Value
You can access error returns from this function through `GetMoviesError` and `GetMoviesStickyError`. See `Error Codes`.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
AlwaysPreview
qtactiontargets
QTKitTimeCode
qttimecode.win
qtwiredspritesjr.win

Declared In
`Movies.h`

SetTrackOffset

Modifies the duration of the empty space that lies at the beginning of a track, thus changing the duration of the entire track.

```
void SetTrackOffset (
   Track theTrack,
   TimeValue movieOffsetTime
);
```

Parameters

theTrack

The track for this operation. Your application obtains this track identifier from such functions as `NewMovieTrack` (page 112) and `GetMovieTrack` (page 85).

movieOffsetTime

The track's offset from the start of the movie, and must be expressed in the time scale of the movie that contains the track.

Return Value

You can access error returns from this function through `GetMoviesError` and `GetMoviesStickyError`. See `Error Codes`.

Discussion

All of the tracks in a movie use the movie's time coordinate system. That is, the movie's time scale defines the basic time unit for each of the movie's tracks. Each track begins at the beginning of the movie, but the track's data might not begin until some time value other than 0. This intervening time is represented by blank space. In an audio track the blank space translates to silence; in a video track the blank space generates no visual image. Each track has its own duration. This duration need not correspond to the duration of the movie. Movie duration always equals the maximum duration of all the tracks.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Movies.h`

SetTrackReference

Modifies an existing track reference.

```
OSErr SetTrackReference (
   Track theTrack,
   Track refTrack,
   OSType refType,
   long index
);
```

Parameters

theTrack

Identifies the track for this operation. Your application obtains this track identifier from such functions as `NewMovieTrack` (page 112) and `GetMovieTrack` (page 85).

refTrack

The track to be identified in the track reference. The toolbox uses this information to update the existing track reference.

refType

The type of reference.

index

The index value of the reference to be changed. You obtain this index value when you create the track reference.

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

Discussion
You may change the track reference so that it identifies a different track in the movie.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`Movies.h`

SetTrackSoundLocalizationSettings

Applies 3D sound effect data to a track.

```
OSErr SetTrackSoundLocalizationSettings (
   Track theTrack,
   Handle settings
);
```

Parameters

theTrack

Identifies the track for this operation. Your application obtains this track identifier from such functions as `NewMovieTrack` (page 112) and `GetMovieTrack` (page 85).

settings

A handle to the settings you want to apply, in the format of an `SSpLocalizationData` record. You can pass a `NIL` handle to indicate that no 3D sound effects should be used for this track. This function makes a copy of the handle passed, so the caller is responsible for disposing of it.

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

Discussion
This function replaces the 3D sound settings for the specified track with the new `SSpLocalizationData` record contained in the `settings` parameter. The effect of the new 3D sound setting takes place immediately. This call always stores the new record passed, even if the track or the computer is not capable of actually meeting the request. When the movie is saved, the 3D sound settings are saved with it.

The following example code shows how to set the static 3D sound setting for a track using this function:

```
// SetTrackSoundLocalizationSettings coding example
void setTrackSoundLocalization(Track t)
{
```

```
    SSpLocalizationData loc;
    Handle h;
    OSErr err;
    loc.cpuLoad =0;
    loc.medium =kSSpMedium_Air;
    loc.humidity =0;
    loc.roomSize =250;
    loc.roomReflectivity =-5;
    loc.reverbAttenuation =-5;
    loc.sourceMode =kSSpSourceMode_Localized;
    loc.referenceDistance =1;
    loc.coneAngleCos =0;
    loc.coneAttenuation =0;
    loc.currentLocation.elevation =0;
    loc.currentLocation.azimuth =0;
    loc.currentLocation.distance =2;
    loc.currentLocation.projectionAngle =0;
    loc.currentLocation.sourceVelocity =0;
    loc.currentLocation.listenerVelocity =0;
    loc.reserved0 =0;
    loc.reserved1 =0;
    loc.reserved2 =0;
    loc.reserved3 =0;
    loc.virtualSourceCount =0;
    err =PtrToHand(&loc, &h, sizeof(loc));
    err =SetTrackSoundLocalizationSettings(t, h);
    DisposeHandle(h);
}
```

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
Movies.h

SetTrackUsage

Specifies whether a track is used in a movie, its preview, its poster, or a combination of these.

```
void SetTrackUsage (
   Track theTrack,
   long usage
);
```

Parameters

theTrack

The track for this operation. Your application obtains this track identifier from such functions as NewMovieTrack (page 112) and GetMovieTrack (page 85).

usage

Contains flags (see below) that specify how the track is to be used. Be sure to set unused flags to 0. See these constants:

`trackUsageInMovie`

`trackUsageInPreview`

`trackUsageInPoster`

Return Value
You can access error returns from this function through `GetMoviesError` and `GetMoviesStickyError`. See `Error Codes`.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
qtinfo
qtinfo.win

Declared In
`Movies.h`

SetTrackVolume

Sets a track's current volume.

```
void SetTrackVolume (
   Track theTrack,
   short volume
);
```

Parameters

theTrack

The track for this operation. Your application obtains this track identifier from such functions as `NewMovieTrack` (page 112) and `GetMovieTrack` (page 85).

volume

The current volume setting of the track represented as a 16-bit, fixed-point number. The high-order 8 bits contain the integer part of the value; the low-order 8 bits contain the fractional part. Volume values range from -1.0 to 1.0. Negative values play no sound but preserve the absolute value of the volume setting. You can use constants (see below) for full volume and no volume. See these constants:

Return Value
You can access error returns from this function through `GetMoviesError` and `GetMoviesStickyError`. See `Error Codes`.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
soundsnippets
soundsnippets.win
vrscript
vrscript.win

Declared In
`Movies.h`

TrackTimeToMediaDisplayTime

Converts a track's time value to a display time value that is appropriate to the track's media, using the track's edit list.

```
TimeValue64 TrackTimeToMediaDisplayTime (
   TimeValue64 value,
   Track theTrack
);
```

Parameters

value
A 64-bit time value that represents the track's time value; it must be expressed in the time scale of the movie that contains the track.

theTrack
A track identifier, which your application obtains from such functions as `NewMovieTrack` (page 112) and `GetMovieTrack` (page 85).

Return Value
A 64-bit time value that represents the corresponding time in media display time, in the media's time coordinate system. If the track time corresponds to empty space, this function returns a value of -1.

Discussion
This function maps the track time through the track's edit list to come up with the media time. This time value contains the track's time value according to the media's time coordinate system. If the time you specified lies outside of the movie's active segment or corresponds to empty space in the track, this function returns a value of -1. Hence you can use it to determine whether a specified track edit is empty.

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

Related Sample Code
MovieVideoChart

Declared In
`Movies.h`

TrackTimeToMediaTime

Converts a track's time value to a time value that is appropriate to the track's media, using the track's edit list.

```
TimeValue TrackTimeToMediaTime (
   TimeValue value,
   Track theTrack
);
```

Parameters

value

The track's time value; must be expressed in the time scale of the movie that contains the track.

theTrack

The track for this operation. Your application obtains this track identifier from such functions as `NewMovieTrack` (page 112) and `GetMovieTrack` (page 85).

Return Value

The track's time value, but in the media's time coordinate system. If the track time corresponds to empty space, this function returns a value of -1.

Discussion

This function maps the track time through the track's edit list to come up with the media time. This time value contains the track's time value according to the media's time coordinate system. If the time you specified lies outside of the movie's active segment or corresponds to empty space in the track, this function returns a value of -1. Hence you can use it to determine whether a specified track edit is empty.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

addhtactions.win

BurntTextSampleCode

qttext

qttext.win

qtwiredactions

Declared In

`Movies.h`

UseMovieEditState

Returns a movie to the condition determined by an edit state created previously.

```
OSErr UseMovieEditState (
   Movie theMovie,
   MovieEditState toState
);
```

Parameters

theMovie

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

toState

The edit state for this operation. Your application obtains this edit state identifier when you create the edit state by calling `NewMovieEditState` (page 112).

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

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`Movies.h`

UseTrackEditState

Returns a track to the condition determined by an edit state created previously.

```
OSErr UseTrackEditState (
   Track theTrack,
   TrackEditState state
);
```

Parameters

theTrack

The track for this operation. Your application obtains this track identifier from such functions as `NewMovieTrack` (page 112) and `GetMovieTrack` (page 85).

state

The edit state for this operation. Your application obtains this edit state identifier when you create the edit state by calling `NewTrackEditState` (page 114).

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

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`Movies.h`

Callbacks

Data Types

DataHandlerComponent

Represents a type used by the Track and Media API.

```
typedef Component DataHandlerComponent;
```

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

Declared In
`Movies.h`

MediaHandlerComponent

Represents a type used by the Track and Media API.

```
typedef Component MediaHandlerComponent;
```

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

Declared In
`Movies.h`

MovieEditState

Represents a type used by the Track and Media API.

```
typedef MovieEditStateRecord * MovieEditState;
```

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

Declared In
`Movies.h`

MovieEditStateRecord

Undocumented

```
struct MovieEditStateRecord {
     long    data[1];
 };
```

Fields
`data`

Discussion
Undocumented

Declared In
`Movies.h`

SampleReference64Ptr

Represents a type used by the Track and Media API.

```
typedef SampleReference64Record * SampleReference64Ptr;
```

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

Declared In
`Movies.h`

SampleReference64Record

Provides a 64-bit version of SampleReferenceRecord.

```
struct SampleReference64Record {
     wide             dataOffset;
     unsigned long    dataSize;
     TimeValue        durationPerSample;
     unsigned long    numberOfSamples;
     short            sampleFlags;
 };
```

Fields
`dataOffset`

Discussion
Specifies the offset into the movie data file. This field specifies the offset into the file of the sample data.

`dataSize`

Discussion
Specifies the total number of bytes of sample data identified by the reference. All samples referenced by a single SampleReference64Record must be the same size.

`durationPerSample`

Discussion
Specifies the duration of each sample in the reference. You must specify this parameter in the media's time scale. All samples referenced by a single SampleReference64Record must be the same duration.

`numberOfSamples`

Discussion
Specifies the number of samples contained in the reference.

`sampleFlag`

Discussion
Contains flags (see below) that control the operation. Set unused flags to 0. See these constants:
`mediaSampleNotSync`

Related Functions
`AddMediaSampleReferences64` (page 28)
`GetMediaSampleReferences64` (page 76)

Declared In
`Movies.h`

SampleReferencePtr

Represents a type used by the Track and Media API.

```
typedef SampleReferenceRecord * SampleReferencePtr;
```

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

Declared In
`Movies.h`

SampleReferenceRecord

Describes a sample or group of similar samples.

```
struct SampleReferenceRecord {
     long         dataOffset;
     long         dataSize;
     TimeValue    durationPerSample;
     long         numberOfSamples;
     short        sampleFlags;
 };
```

Fields
`dataOffset`

Discussion
Specifies the offset into the movie data file. This field specifies the offset into the file of the sample data.

`dataSize`

Discussion
Specifies the total number of bytes of sample data identified by the reference. All samples referenced by a single `SampleReferenceRecord` must be the same size.

`durationPerSample`

Discussion
Specifies the duration of each sample in the reference. You must specify this parameter in the media's time scale. All samples referenced by a single `SampleReferenceRecord` must be the same duration.

`numberOfSamples`

Discussion
Specifies the number of samples contained in the reference.

`sampleFlag`

Discussion
Contains flags (see below) that control the operation. Set unused flags to 0. See these constants:
`mediaSampleNotSync`

Related Functions
`AddMediaSampleReferences` (page 27)
`GetMediaSampleReferences` (page 74)

Declared In
`Movies.h`

TrackEditState

Represents a type used by the Track and Media API.

```
typedef TrackEditStateRecord * TrackEditState;
```

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

Declared In
`Movies.h`

TrackEditStateRecord

Contains a track edit state.

```
struct TrackEditStateRecord {
     long    data[1];
 };
```

Fields
`data`

Discussion
An array of data that constitutes a track edit state.

Declared In
`Movies.h`

Constants

GetMovieImporter Flags

Constants that represent <codeVoice>GetMovieImporter</codeVoice> flags.

```
enum {
  kGetMovieImporterValidateToFind = 1L << 0,
  kGetMovieImporterAllowNewFile = 1L << 1,
  kGetMovieImporterDontConsiderGraphicsImporters = 1L << 2,
  kGetMovieImporterDontConsiderFileOnlyImporters = 1L << 6,
  kGetMovieImporterAutoImportOnly = 1L << 10 /* reject aggressive movie importers
 which have dontAutoFileMovieImport set*/
};
```

Declared In
Movies.h

AddClonedTrackToMovie Values

Constants passed to AddClonedTrackToMovie.

```
enum {
  kQTCloneShareSamples          = 1 << 0,
  kQTCloneDontCopyEdits         = 1 << 1
};
```

Declared In
Movies.h

QTGetMIMETypeInfo Values

Constants passed to QTGetMIMETypeInfo.

```
enum {
  kQTGetMIMETypeInfoIsQuickTimeMovieType = 'moov', /* info is a pointer to a
Boolean*/
  kQTGetMIMETypeInfoIsUnhelpfulType = 'dumb' /* info is a pointer to a Boolean*/
};
```

Declared In
Movies.h

GetMovieIndTrackType Values

Constants passed to GetMovieIndTrackType.

```
enum {
  movieTrackMediaType           = 1 << 0,
  movieTrackCharacteristic      = 1 << 1,
  movieTrackEnabledOnly         = 1 << 2
};
```

Declared In
Movies.h

movieFileSpecValid

Constants grouped with movieFileSpecValid.

```
enum {
  pasteInParallel               = 1 << 0,
  showUserSettingsDialog        = 1 << 1,
  movieToFileOnlyExport         = 1 << 2,
  movieFileSpecValid            = 1 << 3
};
```

Declared In
Movies.h

SetTrackUsage Values

Constants passed to SetTrackUsage.

```
enum {
  trackUsageInMovie             = 1 << 1,
  trackUsageInPreview           = 1 << 2,
  trackUsageInPoster            = 1 << 3
};
```

Declared In
Movies.h

Media Identifiers

Identify media types in QuickTime.

```
enum {
  VideoMediaType                = 'vide',
  SoundMediaType                = 'soun',
  TextMediaType                 = 'text',
  BaseMediaType                 = 'gnrc',
  MPEGMediaType                 = 'MPEG',
  MusicMediaType                = 'musi',
  TimeCodeMediaType             = 'tmcd',
  SpriteMediaType               = 'sprt',
  FlashMediaType                = 'flsh',
  MovieMediaType                = 'moov',
  TweenMediaType                = 'twen',
  ThreeDeeMediaType             = 'qd3d',
  SkinMediaType                 = 'skin',
  HandleDataHandlerSubType      = 'hndl',
  PointerDataHandlerSubType     = 'ptr ',
  NullDataHandlerSubType        = 'null',
  ResourceDataHandlerSubType    = 'rsrc',
  URLDataHandlerSubType         = 'url ',
  AliasDataHandlerSubType       = 'alis',
  WiredActionHandlerType        = 'wire'
};
```

Constants

`SoundMediaType`

Sound channel.

Available in Mac OS X v10.0 and later.

Declared in `Movies.h`.

`TextMediaType`

Text media.

Available in Mac OS X v10.0 and later.

Declared in `Movies.h`.

Declared In

`Movies.h`

Document Revision History

This table describes the changes to *QuickTime Movie Track and Media Reference*.

Date	Notes
2006-11-10	Specify track order numbering.
2006-05-23	New document, based on previously published material, describes the API for managing QuickTime movie tracks and their media.

Document Revision History

Index

I

M

N

O

P

Q

S

T

U