

GetTrackMedia

The `GetTrackMedia` function allows you to determine the media that contains a track's sample data.

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

theTrack Specifies the track for this operation. Your application obtains this track identifier from such Movie Toolbox functions as `NewMovieTrack` and `GetMovieTrack` (described on page 2-151 and page 2-204, respectively).

DESCRIPTION

The `GetTrackMedia` function returns the media identifier that corresponds to the media that specifies the track's sample data. If the function could not locate the media, it sets this returned value to `nil`.

ERROR CODES

<code>invalidTrack</code>	-2009	This track is corrupted or invalid
---------------------------	-------	------------------------------------

GetMediaTrack

The `GetMediaTrack` function allows you to determine the track that uses a specified media.

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

theMedia Specifies the media for this operation. Your application obtains this media identifier from such Movie Toolbox functions as `NewTrackMedia` and `GetTrackMedia` (described on page 2-153 and page 2-206, respectively).

DESCRIPTION

The `GetMediaTrack` function returns the track identifier of the track that uses the media. If the function cannot determine the track that uses the media, it sets this value to `nil`.

ERROR CODES

<code>invalidMedia</code>	-2008	This media is corrupted or invalid
---------------------------	-------	------------------------------------

Working With Alternate Tracks

The Movie Toolbox allows you to define alternate tracks in a movie. You can use alternate tracks to support multiple languages or to present different levels of visual quality in the movie. You collect alternate tracks into groups. Alternate track groups are collections of tracks that conceptually represent some data but are appropriate for use in different play environments. For example, you might have some 4-bit data in one track and some 8-bit data in another. Working with alternate tracks allows you to set up alternatives from which the Movie Toolbox can choose.

The Movie Toolbox selects one track from each alternate group when it plays the movie. For example, you could create a movie that has three separate audio tracks: one in English, one in French, and one in Spanish. You would collect these audio tracks into an alternate group. When the user plays the movie, the Movie Toolbox selects the track from this group that corresponds to the current language setting for the movie.

Similarly, you can use alternate tracks to store data of different quality. When the user plays the movie, the Movie Toolbox selects the track that best suits the capabilities of the Macintosh computer on which the movie is being played. In this manner, you can create a single movie that can accommodate the playback characteristics of a number of different computer configurations.

The Movie Toolbox allows you to store quality information for media structures that are assigned to either sound or video tracks. For all tracks, the Movie Toolbox uses bits 6 and 7 of the quality setting. These bits encode a relative quality value. These values range from 0 to 3. You can use higher quality values to indicate larger sample sizes. For example, consider a movie that has two sound tracks that are alternates for each other—one contains 8-bit sound while the other contains 16-bit sound. You could assign a quality value of `mediaQualityNormal` to the 8-bit media and a value of `mediaQualityBetter` to the 16-bit media. The Movie Toolbox would only play the 16-bit media if the Macintosh configuration could handle 16-bit sound. Otherwise, the Movie Toolbox would use the 8-bit media. The sound media handler determines the sample size for each sound media for the Movie Toolbox by examining the media's sound description structure.

In addition, the Movie Toolbox also uses bits 0 through 5 (the low-order bits) of the quality setting. You use these bits to indicate the pixel depths at which the media should be played. Each bit corresponds to a single depth value, ranging from 1-bit pixels to 32-bit pixels. You may use these bits to control the playback of both video and sound tracks.

As an example, consider a movie that contains three video tracks with the following characteristics:

- | | |
|---------|---|
| Track A | 1-bit video data, no compression |
| Track B | Compressed using the Apple Video Compressor |
| Track C | Compressed using the Joint Photographic Experts Group (JPEG) compressor |

Movie Toolbox

You could assign the following quality values to these track's media structures:

- Track A `mediaQualityDraft` + 1-bit depth + 2-bit depth (quality value is `0x0003: 0x0000 + 0x0003`)
- Track B `mediaQualityNormal` + 4-bit depth + 8-bit depth + 16-bit depth + 32-bit depth (quality value is `0x007C: 0x0040 + 0x003C`)
- Track C `mediaQualityBetter` + 4-bit depth + 8-bit depth + 16-bit depth + 32-bit depth (quality value is `0x00BC: 0x0080 + 0x003C`)

The Movie Toolbox would always use Track A when playing the movie on 1-bit and 2-bit displays. At the other pixel depths, the video media handler determines which track to use by examining the availability and performance of the specified decompressors. If the JPEG decompressor can play back at full frame rate, the Movie Toolbox would use Track C. Otherwise, the Toolbox uses Track B. The video media handler determines the compressor that is appropriate for each media by examining the media's image description structure.

You set a movie's language by calling the `SetMovieLanguage` function.

To establish alternate groups of tracks, you can use the `SetTrackAlternate` and `GetTrackAlternate` functions.

You can work with the language and quality characteristics of media by calling the `GetMediaLanguage`, `SetMediaLanguage`, `GetMediaQuality`, and `SetMediaQuality` functions.

By default, the Movie Toolbox automatically selects the appropriate tracks to play according to a movie's quality and language settings, as well as the capabilities of the Macintosh computer. Whenever your application calls the `SetMovieGWorld`, `SetMovieBox`, `UpdateMovie`, or `SetMovieMatrix` function (described on page 2-159, page 2-161, page 2-126, and page 2-170, respectively), the Movie Toolbox checks each alternate group for an appropriate track. However, you can control this selection process. Use the `SetAutoTrackAlternatesEnabled` function to enable or disable automatic track selection. The `SelectMovieAlternates` function instructs the Movie Toolbox to select appropriate tracks immediately. If no tracks in an alternate track group are enabled, then the Movie Toolbox does not activate any track from that group during automatic track selection.

SetMovieLanguage

The `SetMovieLanguage` function allows your application to specify a movie's language. You specify the language by supplying the appropriate language or region code (see *Inside Macintosh: Text* for more information on language and region codes).

```
pascal void SetMovieLanguage (Movie theMovie, long language);
```

Movie Toolbox

- `theMovie` Specifies the movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie`, `NewMovieFromFile`, and `NewMovieFromHandle` (described on page 2-92, page 2-88, and page 2-90, respectively).
- `language` Specifies the movie’s language or region code.

DESCRIPTION

The Movie Toolbox examines the movie’s alternate groups and selects and enables appropriate tracks. If the Movie Toolbox cannot find an appropriate track, it does not change the movie’s language.

ERROR CODES

- `invalidMovie` -2010 This movie is corrupted or invalid

SelectMovieAlternates

The `SelectMovieAlternates` function allows your application to instruct the Movie Toolbox to select appropriate tracks immediately.

```
pascal void SelectMovieAlternates (Movie theMovie);
```

- `theMovie` Specifies the movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie`, `NewMovieFromFile`, and `NewMovieFromHandle` (described on page 2-92, page 2-88, and page 2-90, respectively).

DESCRIPTION

You can call the `SelectMovieAlternates` function even if you have disabled automatic track selection with the `SetAutoTrackAlternatesEnabled` function (which is described in the next section) or by setting the `newMovieDontAutoAlternate` flag when you created the movie (see page 2-91 for details on this flag).

ERROR CODES

- `invalidMovie` -2010 This movie is corrupted or invalid

SetAutoTrackAlternatesEnabled

The `SetAutoTrackAlternatesEnabled` function allows your application to enable and disable automatic track selection by the Movie Toolbox.

```
pascal void SetAutoTrackAlternatesEnabled (Movie theMovie,
                                           Boolean enable);
```

<code>theMovie</code>	Specifies the movie for this operation. Your application obtains this movie identifier from such functions as <code>NewMovie</code> , <code>NewMovieFromFile</code> , and <code>NewMovieFromHandle</code> (described on page 2-92, page 2-88, and page 2-90, respectively).
<code>enable</code>	Controls automatic track selection. Set this parameter to <code>true</code> to enable automatic track selection. Set this parameter to <code>false</code> to disable automatic track selection.

DESCRIPTION

If automatic track selection is enabled, the Movie Toolbox selects appropriate tracks whenever your application calls the `SetMovieGWorld`, `SetMovieBox`, `UpdateMovie`, or `SetMovieMatrix` functions (described on page 2-159, page 2-161, page 2-126, and page 2-170, respectively). When you enable automatic track selection, the Movie Toolbox immediately selects enabled tracks for the movie. This overrides the setting of the `newMovieDontAutoAlternate` flag (see page 2-91 for details on this flag).

ERROR CODES

<code>invalidMovie</code>	-2010	This movie is corrupted or invalid
---------------------------	-------	------------------------------------

SEE ALSO

You can instruct the Movie Toolbox to select appropriate tracks immediately by calling the `SelectMovieAlternates` function, which is described in the previous section.

SetTrackAlternate

The `SetTrackAlternate` function allows your application to add tracks to or remove tracks from alternate groups.

```
pascal void SetTrackAlternate (Track theTrack, Track alternateT);
```

<code>theTrack</code>	Specifies the track and group for this operation. Your application obtains this track identifier from such Movie Toolbox functions as <code>NewMovieTrack</code> and <code>GetMovieTrack</code> (described on page 2-151 and
-----------------------	--

page 2-204, respectively). The `SetTrackAlternate` function 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 the `alternateT` parameter contains a valid track identifier, the Movie Toolbox adds this track to the group that contains the track specified by the parameter `theTrack`. Note that if the track identified by the parameter `alternateTrack` already belongs to a group, the Movie Toolbox combines the two groups into a single group. Set this parameter to `nil` to remove the track specified by the `theTrack` parameter from its group.

ERROR CODES

<code>invalidTrack</code>	-2009	This track is corrupted or invalid
---------------------------	-------	------------------------------------

SEE ALSO

You can determine all the tracks in a group by calling the `GetTrackAlternate` function, which is described in the next section.

GetTrackAlternate

The `GetTrackAlternate` function allows your application to determine all the tracks in an alternate group. You specify the group by identifying a track in the group. The group list is circular, so you must specify a different track in the group each time you call this function.

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

<code>theTrack</code>	Specifies the track for this operation. Your application obtains this track identifier from such Movie Toolbox functions as <code>NewMovieTrack</code> and <code>GetMovieTrack</code> (described on page 2-151 and page 2-204, respectively).
-----------------------	---

DESCRIPTION

The `GetTrackAlternate` function returns the track identifier of the next track in the group. If the track you specify does not belong to a group, the function returns the same identifier you supply. Because the alternate group list is circular, you have retrieved all the tracks in the group when the function returns the track identifier that you supplied the first time you called the `GetTrackAlternate` function. If there is only one track in an alternate group, this function returns the track identifier you supply.

Movie Toolbox

ERROR CODES

`invalidTrack` -2009 This track is corrupted or invalid

SEE ALSO

You can add a track to a group by calling the `SetTrackAlternate` function, which is described in the previous section.

SetMediaLanguage

The `SetMediaLanguage` function sets a media's language or region code. You should call this function only when you are creating a new media. See *Inside Macintosh: Text* for more information on language and region codes.

```
pascal void SetMediaLanguage (Media theMedia, short language);
```

`theMedia` Specifies the media for this operation. Your application obtains this media identifier from such Movie Toolbox functions as `NewTrackMedia` and `GetTrackMedia` (described on page 2-151 and page 2-204, respectively).

`language` Specifies the media's language or region code.

ERROR CODES

`invalidMedia` -2008 This media is corrupted or invalid

SEE ALSO

You can retrieve a media's language or region code by calling the `GetMediaLanguage` function, which is described in the next section.

GetMediaLanguage

The `GetMediaLanguage` function returns a media's language or region code. See *Inside Macintosh: Text* for more information on language and region codes.

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

`theMedia` Specifies the media for this operation. Your application obtains this media identifier from such Movie Toolbox functions as `NewTrackMedia` and `GetTrackMedia` (described on page 2-153 and page 2-206, respectively).

ERROR CODES

`invalidMedia` -2008 This media is corrupted or invalid

SEE ALSO

You can set a media's language or region code by calling the `SetMediaLanguage` function, which is described in the previous section.

SetMediaQuality

The `SetMediaQuality` function sets a media's quality level value. 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.

```
pascal void SetMediaQuality (Media theMedia, short quality);
```

theMedia Specifies the media for this operation. Your application obtains this media identifier from such Movie Toolbox functions as `NewTrackMedia` and `GetTrackMedia` (described on page 2-153 and page 2-206, respectively).

quality Specifies 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 following bits are defined:

- Bit 0 1 bit per pixel
- Bit 1 2 bits per pixel
- Bit 2 4 bits per pixel
- Bit 3 8 bits per pixel
- Bit 4 16 bits per pixel
- Bit 5 32 bits per pixel

In addition, bits 6 and 7 define the media's quality level. A value of 0 corresponds to the lowest quality level; a value of 3 corresponds to the highest quality level. The following constants define these values:

`mediaQualityDraft`

Specifies the lowest quality level. This constant sets bits 6 and 7 to a value of 0.

`mediaQualityNormal`

Specifies an acceptable quality level. This constant sets bits 6 and 7 to a value of 1.

`mediaQualityBetter`

Specifies a higher quality level. This constant sets bits 6 and 7 to a value of 2.

Movie Toolbox

`mediaQualityBest`

Specifies the highest quality level. This constant sets bits 6 and 7 to a value of 3.

ERROR CODES

`invalidMedia` -2008 This media is corrupted or invalid

SEE ALSO

You can retrieve the quality value of a media by calling the `GetMediaQuality` function, which is described in the next section.

GetMediaQuality

The `GetMediaQuality` function returns a media's quality level value. The Movie Toolbox uses this quality value to influence which track it selects to play on a given Macintosh computer.

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

`theMedia` Specifies the media for this operation. Your application obtains this media identifier from such Movie Toolbox functions as `NewTrackMedia` and `GetTrackMedia` (described on page 2-153 and page 2-206, respectively).

DESCRIPTION

The `GetMediaQuality` function returns 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 following bits are defined:

Bit 0	1 bit per pixel
Bit 1	2 bits per pixel
Bit 2	4 bits per pixel
Bit 3	8 bits per pixel
Bit 4	16 bits per pixel
Bit 5	32 bits per pixel

In addition, bits 6 and 7 define the media's quality level. A value of 0 corresponds to the lowest quality level; a value of 3 corresponds to the highest quality level.

`mediaQualityDraft`

Specifies the lowest quality level. This constant sets bits 6 and 7 to a value of 0.

Movie Toolbox

`mediaQualityNormal`

Specifies an acceptable quality level. This constant sets bits 6 and 7 to a value of 1.

`mediaQualityBetter`

Specifies a higher quality level. This constant sets bits 6 and 7 to a value of 2.

`mediaQualityBest`

Specifies the highest quality level. This constant sets bits 6 and 7 to a value of 3.

ERROR CODES

`invalidMedia` -2008 This media is corrupted or invalid

SEE ALSO

You can set the quality value of a media by calling the `SetMediaQuality` function, which is described in the previous section.

Working With Data References

Media structures identify how and where to find their sample data by means of data references. For sound and video media, **data references** identify files that contain media data; the media data is stored in the data forks of these files. Media handlers use these data references in order to manipulate media data. A single media may contain one or more data references.

Each data reference contains type information that identifies how the reference is specified. Most QuickTime data references use alias information to locate the corresponding files (see *Inside Macintosh: Files* for more information about aliases and the Alias Manager). The type value for data references that use aliases is 'alis'. Note that the Movie Toolbox uses aliases even on Macintosh computers that do not have System 7 installed—your application can use Alias Manager routines if the Movie Toolbox is installed. See “The Movie Toolbox and System 6” on page 2-63 for more information.

The Movie Toolbox identifies a media’s data references with an index value. Index values always range from 1 to the number of references in the media. Data reference indexes provide a convenient way to access each reference in a media.

The Movie Toolbox provides a set of functions that allow you to work with data references. This section describes those functions.

You can use the `GetMediaDataRef` function to retrieve information about a media’s data reference. You can add a data reference to a media by calling the `AddMediaDataRef` function. The `SetMediaRef` function lets you change which file a specified media associates with its data storage.

Your application can determine the number of data references in a media by calling the `GetMediaDataRefCount` function.

AddMediaDataRef

The `AddMediaDataRef` function adds a data reference to a media.

```
pascal OSErr AddMediaDataRef (Media theMedia, short *index,
                              Handle dataRef,
                              OSType dataRefType);
```

<code>theMedia</code>	Specifies the media for this operation. Your application obtains this media identifier from such Movie Toolbox functions as <code>NewTrackMedia</code> and <code>GetTrackMedia</code> (described on page 2-153 and page 2-206, respectively).
<code>index</code>	Contains a pointer to a short integer. The Movie Toolbox returns the index value that is assigned to the new data reference. Your application can use this index to identify the reference to other Movie Toolbox functions, such as <code>GetMediaDataRef</code> (described on page 2-217). If the Movie Toolbox cannot add the data reference to the media, it sets the returned index value to 0.
<code>dataRef</code>	Specifies 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 <code>dataRefType</code> parameter.
<code>dataRefType</code>	Specifies the type of data reference. If the data reference is an alias, you must set this parameter to <code>rAliasType ('alis')</code> , indicating that the reference is an alias. See <i>Inside Macintosh: Files</i> for more information about aliases and the Alias Manager.

ERROR CODES

<code>invalidMedia</code>	-2008	This media is corrupted or invalid
---------------------------	-------	------------------------------------

SetMediaDataRef

The `SetMediaDataRef` function changes the file that the specified media identifies as the location for its data storage.

```
pascal OSErr SetMediaDataRef (Media themedia, short index,
                              Handle dataRef, OSType dataRefType);
```

<code>themedia</code>	Specifies the media for this operation. Your application obtains this media identifier from such Movie Toolbox functions as <code>NewTrackMedia</code> and <code>GetTrackMedia</code> (described on page 2-153 and page 2-206, respectively).
<code>index</code>	Contains a pointer to a short integer. The Movie Toolbox returns the index value that is assigned to the new data reference. Your application can use this index to identify the reference to other Movie Toolbox functions, such

Movie Toolbox

	as <code>GetMediaDataRef</code> (described on page 2-217). As with all data reference functions, the index starts with 1. If the Movie Toolbox cannot add the data reference to the media, it sets the returned index value to 0.
<code>dataRef</code>	Specifies 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 <code>dataRefType</code> parameter.
<code>dataRefType</code>	Specifies the type of data reference. If the data reference is an alias, you must set this parameter to <code>rAliasType ('alis')</code> , indicating that the reference is an alias. See <i>Inside Macintosh: Files</i> for more information about aliases and the Alias Manager.

SPECIAL CONSIDERATIONS

Don't call this function unless you have a really good reason. However, if you want to resolve your own missing data references, or you are developing a special-purpose kind of application, `SetMediaDataRef` may be quite useful.

GetMediaDataRef

The `GetMediaDataRef` function returns a copy of a specified data reference. Your application identifies the data reference with the appropriate data reference index.

```
pascal OSErr GetMediaDataRef (Media theMedia, short index,
                              Handle *dataRef,
                              OSType *dataRefType,
                              long *dataRefattributes);
```

<code>theMedia</code>	Specifies the media for this operation. Your application obtains this media identifier from such Movie Toolbox functions as <code>NewTrackMedia</code> and <code>GetTrackMedia</code> (described on page 2-153 and page 2-206, respectively).
<code>index</code>	Identifies the data reference. You provide the index value that corresponds to the data reference. It must be less than or equal to the value that is returned by the <code>GetMediaDataRefCount</code> function, described in the previous section.
<code>dataRef</code>	Contains a pointer to a field that is to receive a handle to the data reference. The media handler returns a handle to 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 <code>dataRefType</code> parameter. If the function cannot locate the specified data reference, the handler sets this returned value to <code>nil</code> . Set the <code>dataRef</code> parameter to <code>nil</code> if you are not interested in this information.

Movie Toolbox

`dataRefType`

Contains a pointer to a field that is to receive the type of data reference. If the data reference is an alias, the media handler sets this value to 'alias', indicating that the reference is an alias. Set the `dataRefType` parameter to `nil` if you are not interested in this information.

`dataRefAttributes`

Contains a pointer to a field that is to receive the reference's attribute flags. The following flags are available (unused flags are set to 0):

`dataRefSelfReference`

Indicates whether the data reference refers to the movie resource's data file. If this flag is set to 1, the data reference identifies media data that is stored in the same file as the movie resource.

`dataRefWasNotResolved`

Indicates whether the Movie Toolbox resolved the data reference. If this flag is set to 1, the Movie Toolbox could not resolve the data reference. For example, the toolbox may be unable to resolve data references because the required storage device is unavailable at the time a movie is loaded. If the data reference is unresolved, the Movie Toolbox disables the corresponding track.

Set the `dataRefAttributes` parameter to `nil` if you are not interested in this information.

DESCRIPTION

You can use `GetMediaDataRef` function to retrieve information about a data reference. For example, you might want to verify the condition of a movie's data references after loading the movie from its movie file. You could use this function to check each data reference.

ERROR CODES

<code>invalidMedia</code>	-2008	This media is corrupted or invalid
---------------------------	-------	------------------------------------

SEE ALSO

You can add a data reference to a media by calling the `AddMediaDataRef` function, which is described on page 2-216. You must dispose of a media's data references yourself by disposing of its handle. You can determine the number of data references in a media by calling the `GetMediaDataRefCount` function, which is described in the previous section.

GetMediaDataRefCount

The `GetMediaDataRefCount` function allows your application to determine the number of data references in a media.

```
pascal OSErr GetMediaDataRefCount (Media theMedia, short *count);
```

<code>theMedia</code>	Specifies the media for this operation. Your application obtains this media identifier from such Movie Toolbox functions as <code>NewTrackMedia</code> and <code>GetTrackMedia</code> (described on page 2-153 and page 2-206, respectively).
<code>count</code>	Contains a pointer to a field that is to receive the number of data references in the media.

DESCRIPTION

The count of references in a media corresponds to the maximum index value of any reference in the media. You can use this value to control a loop in which you retrieve all of a media's data references, using the `GetMediaDataRef` function, which is described in the next section.

ERROR CODES

<code>invalidMedia</code>	-2008	This media is corrupted or invalid
---------------------------	-------	------------------------------------

Determining Movie Creation and Modification Time

The Movie Toolbox maintains two timestamps in every movie, track, and media. One timestamp, the creation date, indicates the date and time when the item was created. The other, the modification date, contains the date and time when the item was last changed and saved. The timestamp value is in the same format as Macintosh file system creation and modification times; that is, the timestamp indicates the number of seconds since midnight, January 1, 1904.

The Movie Toolbox provides a number of functions that allow your application to retrieve the creation and modification date information from movies, tracks, and media structures. This section describes those functions.

You can use the `GetMovieCreationTime` and `GetMovieModificationTime` functions to work with movie creation and modification dates.

You can use the `GetTrackCreationTime` and `GetTrackModificationTime` functions to retrieve a track's creation and modification dates.

Your application can call the `GetMediaCreationTime` and `GetMediaModificationTime` functions to get a media's creation and modification dates.

GetMovieCreationTime

The `GetMovieCreationTime` function returns a long integer that contains the movie's creation date and time information.

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

theMovie Specifies the movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie`, `NewMovieFromFile`, and `NewMovieFromHandle` (described on page 2-92, page 2-88, and page 2-90, respectively).

ERROR CODES

<code>invalidMovie</code>	-2010	This movie is corrupted or invalid
---------------------------	-------	------------------------------------

GetMovieModificationTime

The `GetMovieModificationTime` function returns a movie's modification date.

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

theMovie Specifies the movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie`, `NewMovieFromFile`, and `NewMovieFromHandle` (described on page 2-92, page 2-88, and page 2-90, respectively).

DESCRIPTION

The `GetMovieModificationTime` function returns a long integer that contains the movie's modification date and time information.

ERROR CODES

<code>invalidMovie</code>	-2010	This movie is corrupted or invalid
---------------------------	-------	------------------------------------

GetTrackCreationTime

The `GetTrackCreationTime` function returns a track's creation date.

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

Movie Toolbox

`theTrack` Specifies the track for this operation. Your application obtains this track identifier from such Movie Toolbox functions as `NewMovieTrack` and `GetMovieTrack` (described on page 2-151 and page 2-204, respectively).

DESCRIPTION

The `GetTrackCreationTime` function returns a long integer that contains the track's creation date and time information.

ERROR CODES

`invalidTrack` -2009 This track is corrupted or invalid

GetTrackModificationTime

The `GetTrackModificationTime` function returns a track's modification date.

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

`theTrack` Specifies the track for this operation. Your application obtains this track identifier from such Movie Toolbox functions as `NewMovieTrack` and `GetMovieTrack` (described on page 2-151 and page 2-204, respectively).

DESCRIPTION

The `GetTrackModificationTime` function returns a long integer that contains the track's modification date and time information.

ERROR CODES

`invalidTrack` -2009 This track is corrupted or invalid

GetMediaCreationTime

The `GetMediaCreationTime` function returns the creation date stored in the media.

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

`theMedia` Specifies the media for this operation. Your application obtains this media identifier from such Movie Toolbox functions as `NewTrackMedia` and `GetTrackMedia` (described on page 2-153 and page 2-206, respectively).

Movie Toolbox

DESCRIPTION

The `GetMediaCreationTime` function returns a long integer that contains the media's creation date and time information.

ERROR CODES

<code>invalidMedia</code>	-2008	This media is corrupted or invalid
---------------------------	-------	------------------------------------

GetMediaModificationTime

The `GetMediaModificationTime` function returns a media's modification date.

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

<code>theMedia</code>	Specifies the media for this operation. Your application obtains this media identifier from such Movie Toolbox functions as <code>NewTrackMedia</code> and <code>GetTrackMedia</code> (described on page 2-153 and page 2-206, respectively).
-----------------------	---

DESCRIPTION

The `GetMediaModificationTime` function returns a long integer that contains the media's modification date and time information.

ERROR CODES

<code>invalidMedia</code>	-2008	This media is corrupted or invalid
---------------------------	-------	------------------------------------

Working With Media Samples

The Movie Toolbox provides a number of functions that allow applications to determine information about a movie's sample data. This section discusses these functions. Refer to "Adding Samples to Media Structures" beginning on page 2-271 for information about functions that allow you to retrieve sample data from a media.

Your application can use the `GetMovieDataSize`, `GetTrackDataSize`, and `GetMediaDataSize` functions to determine the size, in bytes, of the data stored in a media, movie, or track.

You can use the `GetMediaSampleDescriptionCount` and `GetMediaSampleDescription` functions to retrieve a media's sample descriptions. The `SetMediaSampleDescription` function enables you to change the contents of a particular sample description associated with a media. The `GetMediaSampleCount` function determines the number of samples in a media. The `SampleNumToMediaTime` and `MediaTimeToSampleNum` functions allow you to convert from a time value to a sample number and vice versa. You can use the functions described in "Finding Interesting Times" beginning on page 2-196 to locate specific samples in a media.

GetMovieDataSize

The `GetMovieDataSize` function allows your application to determine the size, in bytes, of the sample data in a segment of a movie.

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

theMovie Specifies the movie for this operation. You obtain this movie identifier from such functions as `NewMovie`, `NewMovieFromFile`, and `NewMovieFromHandle` (described on page 2-92, page 2-88, and page 2-90, respectively).

startTime Contains a time value specifying the starting point of the segment.

duration Contains a time value that specifies the duration of the segment.

DESCRIPTION

The `GetMovieDataSize` function returns a long integer that contains the size, in bytes, of the movie's sample data that lies in the specified segment. `GetMovieDataSize` counts each use of a sample. That is, if a movie 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 movie's sample data, and corresponds to the amount of movie data that will be retrieved when you call the `FlattenMovie` function or `FlattenMovieData` function (described on page 2-105 and page 2-107, respectively).

ERROR CODES

<code>invalidMovie</code>	-2010	This movie is corrupted or invalid
<code>invalidDuration</code>	-2014	This duration value is invalid
<code>invalidTime</code>	-2015	This time value is invalid

GetTrackDataSize

The `GetTrackDataSize` function allows your application to determine the size, in bytes, of the sample data in a segment of a track.

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

theTrack Specifies the track for this operation. You obtain this track identifier from such Movie Toolbox functions as `NewMovieTrack` and `GetMovieTrack` (described on page 2-151 and page 2-204, respectively).

Movie Toolbox

`startTime` Contains a time value specifying the starting point of the segment.
`duration` Contains a time value that specifies the duration of the segment.

DESCRIPTION

The `GetTrackDataSize` function returns a long integer that contains the size, in bytes, of the track's sample data that lies in the specified segment.

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.

ERROR CODES

<code>invalidTrack</code>	-2009	This track is corrupted or invalid
<code>invalidDuration</code>	-2014	This duration value is invalid
<code>invalidTime</code>	-2015	This time value is invalid

GetMediaDataSize

The `GetMediaDataSize` function allows your application to determine the size, in bytes, of the sample data in a media segment.

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

`theMedia` Specifies the media for this operation. Your application obtains this media identifier from such Movie Toolbox functions as `NewTrackMedia` and `GetTrackMedia` (described on page 2-153 and page 2-206, respectively).
`startTime` Contains a time value specifying the starting point of the segment.
`duration` Contains a time value that specifies the duration of the segment.

DESCRIPTION

The `GetMediaDataSize` function returns a long integer that contains the size, in bytes, of the media's sample data that lies in the specified segment. Note that this number does not necessarily correspond to the amount of sample data used in the track that contains the media. Some samples in the media may not be used in the track, and others may be used more than once.

ERROR CODES

<code>invalidMedia</code>	-2008	This media is corrupted or invalid
<code>invalidDuration</code>	-2014	This duration value is invalid
<code>invalidTime</code>	-2015	This time value is invalid

GetMediaSampleCount

The `GetMediaSampleCount` function allows you to determine the number of samples in a media.

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

`theMedia` Specifies the media for this operation. You obtain this media identifier from such Movie Toolbox functions as `NewTrackMedia` and `GetTrackMedia` (described on page 2-153 and page 2-206, respectively).

DESCRIPTION

The `GetMediaSampleCount` function returns a long integer that contains the number of samples in the specified media. Note that this number does not necessarily correspond to the number of samples used in the track that contains the media. Some samples in the media may not be used in the track, and others may be used more than once.

ERROR CODES

`invalidMedia` -2008 This media is corrupted or invalid

GetMediaSampleDescriptionCount

The `GetMediaSampleDescriptionCount` function returns the number of sample descriptions in a media.

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

`theMedia` Specifies the media for this operation. You obtain this media identifier from such Movie Toolbox functions as `NewTrackMedia` and `GetTrackMedia` (described on page 2-153 and page 2-206, respectively).

DESCRIPTION

The Movie Toolbox identifies a media's sample descriptions with an index value. Index values always range 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.

The format of sample descriptions differs by media type. Sample descriptions for image data are defined by image description structures, which are discussed in the chapter "Image Compression Manager" in this book. Sample descriptions for sound are defined by sound description structures, which are discussed in "The Sound Description Structure" beginning on page 2-79. Sample descriptions for text are defined by text

description structures, which are described in “Text Media Handler Functions” beginning on page 2-290.

ERROR CODES

`invalidMedia` -2008 This media is corrupted or invalid

SEE ALSO

You can use the value returned by this function to control a loop in which you retrieve each sample description in a media by calling the `GetMediaSampleDescription` function, which is described in the next section.

GetMediaSampleDescription

The `GetMediaSampleDescription` function allows you to retrieve a sample description from a media.

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

<code>theMedia</code>	Specifies the media for this operation. You obtain this media identifier from such Movie Toolbox functions as <code>NewTrackMedia</code> and <code>GetTrackMedia</code> (described on page 2-153 and page 2-206, respectively).
<code>index</code>	Specifies the index of the sample description to retrieve. This index corresponds to the sample description itself, not the samples in the media.
<code>descH</code>	Specifies a handle that is to receive the sample description. The Movie Toolbox correctly resizes this handle for the returned sample description. If there is no description for the specified index, the function returns this handle unchanged. Your application must allocate and dispose of this handle.

DESCRIPTION

This function provides a convenient way to retrieve information that describes a sample. For example, you can use this function to retrieve an image media’s color lookup table.

The format of sample descriptions differs by media type. Sample descriptions for image data are defined by image description structures, which are discussed in the chapter “Image Compression Manager” in this book. Sample descriptions for sound are defined by sound description structures, which are discussed earlier in this chapter. Sample descriptions for text are defined by text description data structures, which are described in “Text Media Handler Functions” beginning on page 2-290.

Movie Toolbox

The Movie Toolbox identifies a media’s sample descriptions with an index value. Index values always range 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.

ERROR CODES

<code>invalidMedia</code>	-2008	This media is corrupted or invalid
<code>badDataRefIndex</code>	-2050	Data reference index value is invalid
Memory Manager errors		

SEE ALSO

You can determine the number of sample descriptions in a media by calling the `GetMediaSampleDescriptionCount` function, which is described in the previous section.

SetMediaSampleDescription

The `SetMediaSampleDescription` function lets you change the contents of a particular sample description of a specified media.

```
pascal OSErr SetMediaSampleDescription (Media theMedia,
                                         long index,
                                         SampleDescriptionHandle descH);
```

<code>theMedia</code>	Specifies the media for this operation. You obtain this media identifier from such Movie Toolbox functions as <code>NewTrackMedia</code> and <code>GetTrackMedia</code> (described on page 2-153 and page 2-206, respectively).
<code>index</code>	Specifies the index of the sample description to be changed. This index corresponds to the sample description itself, not the samples in the media. This long integer must be between 1 and the largest sample description index.
<code>descH</code>	Specifies the handle to the sample description. If there is no description for the specified index, the function returns this handle unchanged.

DESCRIPTION

The `SetMediaSampleDescription` function can be useful in the case of a media handler, such as a text media handler, that stores playback information in its sample description, as opposed to just data format information (as in the case of the video media handler). For more on media handlers, see *Inside Macintosh: QuickTime Components*.

SPECIAL CONSIDERATIONS

Because a sample description structure may define the format of the data, you should not assume the description describes the data. You should use this function only on an inactive track.

MediaTimeToSampleNum

The `MediaTimeToSampleNum` function allows you to find the sample that contains the data for a specified time. You indicate the time in the media's time scale.

```
pascal void MediaTimeToSampleNum (Media theMedia, TimeValue time,
                                   long *sampleNum,
                                   TimeValue *sampleTime,
                                   TimeValue *sampleDuration);
```

<code>theMedia</code>	Specifies the media for this operation. You obtain this media identifier from such Movie Toolbox functions as <code>NewTrackMedia</code> and <code>GetTrackMedia</code> (described on page 2-153 and page 2-206, respectively).
<code>time</code>	Specifies the time for which you are retrieving sample information. You must specify this value in the media's time scale.
<code>sampleNum</code>	Contains 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 <code>time</code> parameter.
<code>sampleTime</code>	Contains a pointer to a time value. The <code>MediaTimeToSampleNum</code> function updates this time value to indicate the starting time of the sample that contains data for the time specified by the <code>time</code> parameter. This time value is expressed in the media's time scale. Set this parameter to <code>nil</code> if you do not want this information.
<code>sampleDuration</code>	Contains a pointer to a time value. The Movie Toolbox returns the duration of the sample that contains data for the time specified by the <code>time</code> parameter. This time value is expressed in the media's time scale. Set this parameter to <code>nil</code> if you do not want this information.

DESCRIPTION

The Movie Toolbox returns information about the sample that contains data for that time, including its starting time, duration, and sample number.

The `MediaTimeToSampleNum` function does not account for edits applied to the media by a movie's tracks. If you want to work with edits, use the functions that allow you to look for interesting times. These functions are described in "Finding Interesting Times," beginning on page 2-196.

ERROR CODES

`invalidMedia` `-2008` This media is corrupted or invalid

SEE ALSO

You can convert a sample number into a time in a media's time scale by calling the `SampleNumToMediaTime` function, which is described in the next section.

SampleNumToMediaTime

The `SampleNumToMediaTime` function allows you to find the time at which a specified sample plays. This time is expressed in the media's time scale.

```
pascal void SampleNumToMediaTime (Media theMedia,
                                   long logicalSampleNum,
                                   TimeValue *sampleTime,
                                   TimeValue *sampleDuration);
```

`theMedia` Specifies the media for this operation. You obtain this media identifier from such Movie Toolbox functions as `NewTrackMedia` and `GetTrackMedia` (described on page 2-153 and page 2-206, respectively).

`logicalSampleNum`
Specifies the sample number.

`sampleTime`
Contains a pointer to a time value. The `MediaTimeToSampleNum` 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 do not want this information.

`sampleDuration`
Contains 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 do not want this information.

ERROR CODES

`invalidMedia` `-2008` This media is corrupted or invalid

SEE ALSO

You can find the sample for a specified time by calling the `MediaTimeToSampleNum` function, which is described in the previous section.

Working With Movie User Data

Each movie, track, and media can contain a user data list, which your application can use in any way you want. A **user data list** contains all the user data for a movie, track, or media. Each user data list may contain one or more **user data items**. All QuickTime user data items share several attributes.

First, each user data item carries a type identifier. This type is similar to a Resource Manager resource type, and is stored in a long integer. Apple has reserved all lowercase user data type values. You are free to create user data type values using uppercase letters. Apple recommends using type values that begin with the © character (Option-G) to specify user data items that store text data.

The following user data types are currently defined:

'©nam'	Movie's name
'©cpy'	Copyright statement
'©day'	Date the movie content was created
'©dir'	Name of movie's director
'©ed1' to '©ed9'	Edit dates and descriptions
'©fmt'	Indication of movie format (computer-generated, digitized, and so on)
'©inf'	Information about the movie
'©prd'	Name of movie's producer
'©prf'	Names of performers
'©req'	Special hardware and software requirements
'©src'	Credits for those who provided movie source content
'©wrt'	Name of movie's writer

User data items of these types must contain text data only.

Second, the Movie Toolbox allows you to create more than one user data item in a user data list. Therefore, each user data item is identified by a unique index. Index values are assigned sequentially within a user data type and start at 1.

Finally, you may create alternate text for a given user data text item. For example, you may want to support multiple languages and may therefore want to create different text for each language. The Movie Toolbox allows you to specify different versions of the text of a single user data item. These versions are distinguished by their region code values.

The Movie Toolbox provides a number of functions that allow you to work with user data. Before you can work with the contents of a user data list, you must obtain a reference to the list. The `GetMovieUserData`, `GetTrackUserData`, or `GetMediaUserData` functions allow you to get a reference to a user data list. You can then use the `GetUserData`, `AddUserData`, and `RemoveUserData` functions to work with the items contained in the user data list. If your user data items contain text data, you can use the `AddUserDataText`, `GetUserDataText`, and `RemoveUserDataText` functions to work with the text of a user data item. Note that a single user data item can store either text or other data, but not both.

Movie Toolbox

You can count the number of user data items of a specified type in a movie, track, or media by calling the `CountUserDataOfType` function. You can use the `GetNextUserDataOfType` function to scan all the types of user data in a specified user data list.

The Movie Toolbox also supplies a number of functions for the manipulation of user data. The `SetUserDataItem` and `GetUserDataItem` functions allow easy access to data stored in user data items. The `NewUserData` and `DisposeUserData` functions provide for the use of user data outside of the immediate context of QuickTime movies. Your applications and components can also create user data structures. The `PutUserDataIntoHandle` and the `NewUserDataFromHandle` functions permit user data to be stored and retrieved in a manner similar to public movies (also called *atoms*). See the chapter “Movie Resource Formats” in this book for details on atoms.

GetMovieUserData

The `GetMovieUserData` function allows your application to obtain access to a movie’s user data list. You can then use the `GetUserData`, `AddUserData`, and `RemoveUserData` functions (described on page 2-235, page 2-235, and page 2-236, respectively) to manipulate the contents of the user data list. If the data list contains text data, you can use the `GetUserDataText`, `AddUserDataText`, and `RemoveUserDataText` functions (described on page 2-237, page 2-236, and page 2-238, respectively) to work with its contents.

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

theMovie Specifies the movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie`, `NewMovieFromFile`, and `NewMovieFromHandle` (described on page 2-92, page 2-88, and page 2-90, respectively).

DESCRIPTION

The `GetMovieUserData` function returns a reference to the movie’s user data list. This reference is valid until you dispose of the movie. When you save the movie, the Movie Toolbox saves the user data as well. If the function could not locate the movie’s user data, it sets this returned value to `nil`.

ERROR CODES

invalidMovie -2010 This movie is corrupted or invalid
Memory Manager errors

SEE ALSO

You can use the `GetMediaUserData` function (described on page 2-233) to gain access to a media's user data. Similarly, you can use the `GetTrackUserData` function (described in the next section) to work with a track's user data.

GetTrackUserData

The `GetTrackUserData` function allows your application to obtain access to a track's user data list. You can then use the `GetUserData`, `AddUserData`, and `RemoveUserData` functions (described on page 2-235, page 2-235, and page 2-236, respectively) to manipulate the contents of the user data list. If the data list contains text data, you can use the `GetUserDataText`, `AddUserDataText`, and `RemoveUserDataText` functions (described on page 2-237, page 2-236, and page 2-238, respectively) to work with its contents.

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

theTrack Specifies the track for this operation. Your application obtains this track identifier from such Movie Toolbox functions as `NewMovieTrack` and `GetMovieTrack` (described on page 2-151 and page 2-204, respectively).

DESCRIPTION

The `GetTrackUserData` function returns a reference to the track's user data list. This reference is valid until you dispose of the track. When you save the track, the Movie Toolbox saves the user data as well. If the function could not locate the track's user data, it sets this returned value to `nil`.

ERROR CODES

<code>invalidTrack</code>	-2009	This track is corrupted or invalid
Memory Manager errors		

SEE ALSO

You can use the `GetMediaUserData` function to gain access to a media's user data (described on page 2-233). Similarly, you can use the `GetMovieUserData` function (described on page 2-231) to work with a movie's user data.

GetMediaUserData

The `GetMediaUserData` function allows your application to obtain access to a media's user data list. You can then use the `GetUserData`, `AddUserData`, and `RemoveUserData` functions (described on page 2-235, page 2-235, and page 2-236, respectively) to manipulate the contents of the user data list. If the data list contains text data, you can use the `GetUserDataText`, `AddUserDataText`, and `RemoveUserDataText` functions (described on page 2-237, page 2-236, and page 2-238, respectively) to work with its contents.

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

theMedia Specifies the media for this operation. Your application obtains this media identifier from such Movie Toolbox functions as `NewTrackMedia` and `GetTrackMedia` (described on page 2-153 and page 2-206, respectively).

DESCRIPTION

The `GetMediaUserData` function returns a reference to the media's user data list. This reference is valid until you dispose of the media. When you save the media, the Movie Toolbox saves the user data as well. If the function could not locate the media's user data, it sets this returned value to `nil`.

ERROR CODES

`invalidMedia` -2008 This media is corrupted or invalid
Memory Manager errors

SEE ALSO

You can use the `GetMovieUserData` function to gain access to a movie's user data (described on page 2-231). Similarly, you can use the `GetTrackUserData` function (described in the previous section) to work with a track's user data.

GetNextUserDataTypes

The `GetNextUserDataTypes` function allows you to retrieve the next user data type in a specified user data list. You can use this function to scan all the user data types in a user data list.

```
pascal long GetNextUserDataTypes (UserData theUserData,
                                   OSType udType);
```

Movie Toolbox

`theUserData`

Specifies the user data list for this operation. You obtain this list reference by calling the `GetMovieUserData`, `GetTrackUserData`, or `GetMediaUserData` function (described on page 2-231, page 2-232, and page 2-233, respectively).

`udType`

Specifies a user data type. Set this parameter to 0 to retrieve the first user data type in the user data list. On subsequent requests, use the previous value returned by this function.

DESCRIPTION

The `GetNextUserDataTypes` function returns an operating-system data type containing the next user data type value in the specified user data list. When you reach the end of the user data list, this function sets the returned value to 0. You can use this value to stop your scanning loop.

ERROR CODES

None

CountUserDataTypes

The `CountUserDataTypes` function allows you to determine the number of items of a given type in a user data list.

```
pascal short CountUserDataTypes (UserData theUserData,
                                OSType udType);
```

`theUserData`

Specifies the user data list for this operation. You obtain this list reference by calling the `GetMovieUserData`, `GetTrackUserData`, or `GetMediaUserData` function (described on page 2-231, page 2-232, and page 2-233, respectively).

`udType`

Specifies the type. The Movie Toolbox determines the number of items of this type in the user data list.

DESCRIPTION

The `CountUserDataTypes` function returns a short integer that contains the number of items of the specified type in the user data list.

ERROR CODES

None

AddUserData

The `AddUserData` function allows your application to add an item to a user data list. You specify the user data list, the data to be added, and the data's type value.

```
pascal OSErr AddUserData (UserData theUserData,
                          Handle data, OSType udType);
```

<code>theUserData</code>	Specifies the user data list for this operation. You obtain this item reference by calling the <code>GetMovieUserData</code> , <code>GetTrackUserData</code> , or <code>GetMediaUserData</code> function (described on page 2-231, page 2-232, and page 2-233, respectively).
<code>data</code>	Contains a handle to the data to be added to the user data list.
<code>udType</code>	Specifies the type that is to be assigned to the new item.

DESCRIPTION

The Movie Toolbox places the specified data into the user data and assigns an index value that identifies the new item.

ERROR CODES

Memory Manager errors

GetUserData

The `GetUserData` function returns a specified user data item.

```
pascal OSErr GetUserData (UserData theUserData, Handle data,
                          OSType udType, long index);
```

<code>theUserData</code>	Specifies the user data list for this operation. You obtain this list reference by calling the <code>GetMovieUserData</code> , <code>GetTrackUserData</code> , or <code>GetMediaUserData</code> function (described on page 2-231, page 2-232, and page 2-233, respectively).
<code>data</code>	Contains a handle that is to receive the data from the specified item. The <code>GetUserData</code> function resizes this handle as appropriate to accommodate the item. Your application is responsible for releasing this handle when you are done with it. Set this parameter to <code>nil</code> if you do not want to retrieve the user data item. This can be useful if you want to verify that a user data item exists, but you do not need to work with the item's contents.
<code>udType</code>	Specifies the item's type value.

Movie Toolbox

index Specifies the item's index value. This parameter must specify an item in the user data list identified by the parameter `theUserData`.

ERROR CODES

`userDataItemNotFound` -2026 Cannot locate this user data item
Memory Manager errors

RemoveUserData

The `RemoveUserData` function removes an item from a user data list. After the Movie Toolbox removes the item, it rennumbers the remaining items of that type so that the index values are sequential and start at 1.

```
pascal OSErr RemoveUserData (UserData theUserData, OSType udType,
                             long index);
```

theUserData

Specifies the user data list for this operation. You obtain this list reference by calling the `GetMovieUserData`, `GetTrackUserData`, or `GetMediaUserData` function (described on page 2-231, page 2-232, and page 2-233, respectively).

udType Specifies the item's type value.

index Specifies the item's index value. This parameter must specify an item in the user data list identified by the parameter `theUserData`.

ERROR CODES

`userDataItemNotFound` -2026 Cannot locate this user data item
Memory Manager errors

AddUserDataText

The `AddUserDataText` function allows your application to place language-tagged text into an item in a user data list. You specify the user data list and item, the data to be added, the data's type value, and the language code of the data.

```
pascal OSErr AddUserDataText (UserData theUserData, Handle data,
                              OSType udType, long index,
                              short itlRegionTag);
```

Movie Toolbox

`theUserData`

Specifies the user data list for this operation. You obtain this list reference by calling the `GetMovieUserData`, `GetTrackUserData`, or `GetMediaUserData` function (described on page 2-231, page 2-232, and page 2-233, respectively).

`data`

Contains a handle to the data to be added to the user data list.

`udType`

Specifies the type that is to be assigned to the new item.

`index`

Specifies the item to which the text is to be added. This parameter must specify an item in the user data list identified by the parameter `theUserData`.

`itlRegionTag`

Specifies the region code of the text to be added. If there is already text with this region code in the item, the function replaces the existing text with the data specified by the `data` parameter. See *Inside Macintosh: Text* for more information about language and region codes.

DESCRIPTION

The Movie Toolbox places the specified data into the user data item. If the item does not exist when you call this function, the Movie Toolbox creates a new item for you (this is true only if the item you are adding is the first item in the list; otherwise, you must create the item yourself).

ERROR CODES

`userDataItemNotFound` -2026 Cannot locate this user data item
Memory Manager errors

GetUserDataText

The `GetUserDataText` function allows your application to retrieve language-tagged text from an item in a user data list. You specify the user data list and item, and the item's type value and language code. The Movie Toolbox retrieves the specified text from the user data item.

```
pascal OSErr GetUserDataText (UserData theUserData, Handle data,
                               OSType udType, long index,
                               short itlRegionTag);
```

`theUserData`

Specifies the user data list for this operation. You obtain this list reference by calling the `GetMovieUserData`, `GetTrackUserData`, or `GetMediaUserData` function (described on page 2-231, page 2-232, and page 2-233, respectively).

Movie Toolbox

<code>data</code>	Contains a handle that is to receive the data. The <code>GetUserDataText</code> function resizes this handle as appropriate. Your application must dispose of the handle when you are done with it.
<code>udType</code>	Specifies the item's type value.
<code>index</code>	Specifies the item's index value. This parameter must specify an item in the user data list identified by the parameter <code>theUserData</code> .
<code>itlRegionTag</code>	Specifies the language code of the text to be retrieved. See <i>Inside Macintosh: Text</i> for more information about language and region codes.

ERROR CODES

<code>userDataItemNotFound</code>	-2026	Cannot locate this user data item
Memory Manager errors		

RemoveUserDataText

The `RemoveUserDataText` function allows your application to remove language-tagged text from an item in a user data list. You specify the user data list and item, and the item's type value and language code. The Movie Toolbox removes the specified text from the user data item.

```
pascal OSErr RemoveUserDataText (UserData theUserData,
                                OSType udType, long index,
                                short itlRegionTag);
```

<code>theUserData</code>	Specifies the user data list for this operation. You obtain this list reference by calling the <code>GetMovieUserData</code> , <code>GetTrackUserData</code> , or <code>GetMediaUserData</code> function (described on page 2-231, page 2-232, and page 2-233, respectively).
<code>udType</code>	Specifies the item's type value.
<code>index</code>	Specifies the item's index value. This parameter must specify an item in the user data list identified by the parameter <code>theUserData</code> .
<code>itlRegionTag</code>	Specifies the language code of the text to be removed. See <i>Inside Macintosh: Text</i> for more information about language and region codes.

ERROR CODES

<code>userDataItemNotFound</code>	-2026	Cannot locate this user data item
Memory Manager errors		

SetUserDataItem

The `SetUserDataItem` allows your application to set an item in a user data list. You specify the user data list, the data to be set, the size of the data to be set, and the data's type value.

```
pascal OSErr SetUserDataItem (UserData theUserData,
                               void *data, long size, long udType,
                               long index);
```

`theUserData`

Specifies the user data list for this operation. You obtain this item reference by calling the `GetMovieUserData`, `GetTrackUserData`, or `GetMediaUserData` function (described on page 2-231, page 2-232, and page 2-233, respectively).

`data`

Contains a pointer to the data item to be set in a user data list.

`size`

Specifies the size of the information pointed to by the `data` parameter.

`udType`

Specifies the type value assigned to the new item.

`index`

Specifies the item's index value. This parameter must specify an item in the user data list identified by the parameter `theUserData`. An index value of 0 or 1 implies the first item, which is created if it doesn't already exist.

DESCRIPTION

You must provide the size of the information specified in the `data` parameter because the data may be embedded inside a larger data structure or may be on the stack.

SPECIAL CONSIDERATIONS

The data pointer must be locked, since `SetUserDataItem` may move memory.

SEE ALSO

The `SetUserDataItem` function is a pointer-based version of `AddUserData`, which is described on page 2-235.

ERROR CODES

Memory Manager errors

GetUserDataItem

The `GetUserDataItem` function returns a specified user data item. `GetUserDataItem` is a pointer-based version of the `GetUserData` function, which is described on page 2-235.

```
pascal OSErr GetUserDataItem (UserData theUserData,
                             void *data, long size,
                             OSType udType, long index);
```

`theUserData`

Specifies the user data list for this operation. You obtain this list reference by calling the `GetMovieUserData`, `GetTrackUserData`, or `GetMediaUserData` function (described on page 2-231, page 2-232, and page 2-233, respectively).

`data`

Contains a pointer that is to receive the data from the specified item.

`size`

Specifies the size of the item.

`udType`

Specifies the item's type value.

`index`

Specifies the item's index value. This parameter must specify an item in the user data list identified by the parameter `theUserData`.

DESCRIPTION

If the `size` field provided doesn't match the exact size of the actual user data item, an error is returned. In this case, you should use `GetUserData` instead.

`GetUserDataItem` is useful for retrieving small, fixed-size pieces of user data without having to create a handle. You can pass 0 or 1 for the `index` parameter to indicate the first item.

ERROR CODES

<code>userDataItemNotFound</code>	-2026	Cannot locate this user data item
Memory Manager errors		

NewUserData

The `NewUserData` function creates a new user data structure.

```
pascal OSErr NewUserData (UserData *theUserData);
```

`theUserData`

Contains a pointer to the user data structure.

DESCRIPTION

You can manipulate the user data structure with any of the standard user data functions described in “Working With Movie User Data” beginning on page 2-230. If the `NewUserData` function fails, the parameter `theUserData` is set to `nil`.

ERROR CODES

<code>memFullErr</code>	<code>-108</code>	Not enough room in heap zone
-------------------------	-------------------	------------------------------

DisposeUserData

The `DisposeUserData` function disposes of a user data structure created by the `NewUserData` function.

```
pascal OSErr DisposeUserData (UserData theUserData);
```

`theUserData`

Specifies the user data structure that is to be disposed of. It is acceptable but unnecessary to pass `nil` in the parameter `theUserData`.

DESCRIPTION

You should call `DisposeUserData` only on a user data structure that you have allocated.

SPECIAL CONSIDERATIONS

Don't dispose of user data references obtained from the Movie Toolbox function `GetMovieUserData`, `GetTrackUserData`, or `GetMediaUserData` (described on page 2-231, page 2-232, and page 2-233, respectively).

PutUserDataIntoHandle

The `PutUserDataIntoHandle` function takes a specified user data structure and replaces the contents of the handle with a publicly parseable form of the user data.

```
pascal OSErr PutUserDataIntoHandle (UserData theUserData,
                                     Handle h);
```

`theUserData`

Specifies the user data structure that is to be disposed of.

`h`

Contains a handler to the user data structure specified in the parameter `theUserData`.

DESCRIPTION

The contents of the `h` parameter are appropriate for storage as an atom, much like a public movie. See the chapter “Movie Resource Formats” in this book for details on the QuickTime atoms.

NewUserDataFromHandle

The `NewUserDataFromHandle` function creates a new user data structure from a handle.

```
pascal OSErr NewUserDataFromHandle (Handle h,
                                     UserData *theUserData);
```

`h` Contains a handle to the data structure specified in the parameter `theUserData`.

`theUserData` Contains a pointer to a new user data structure.

DESCRIPTION

The handle specified in the `h` parameter must be in the standard user data storage format (that is, as an atom, just like a public movie). Usually the handle will have been created by calling `PutUserDataIntoHandle` (described in the previous section).

ERROR CODES

`memFullErr` -108 Not enough room in heap zone

Functions for Editing Movies

The Movie Toolbox provides a number of functions that allow applications to edit existing movies or create the contents of new movies. This section describes those functions. It has been divided into the following topics:

- “Editing Movies” describes a number of functions that work with the current movie selection, supporting such user operations as cut, copy, and paste
- “Undo for Movies” discusses the functions that your application can use to support an undo capability for movie editing
- “Low-Level Movie-Editing Functions” discusses several functions that allow your application to perform detailed editing on movies
- “Editing Tracks” describes functions that your application can use to edit the contents of tracks

Movie Toolbox

- “Undo for Tracks” discusses the functions that your application can use to support an undo capability for track editing
- “Adding Samples to Media Structures” describes the Movie Toolbox functions that allow you to edit media

Editing Movies

The Movie Toolbox provides a set of high-level functions that allow you to edit movies. This section describes these high-level editing functions. These functions work with a movie’s current selection. The current selection is defined by a starting time and a duration.

The Movie Toolbox also provides functions that allow you to edit movie segments. Those functions are described in “Low-Level Movie-Editing Functions” beginning on page 2-257.

The movies created by these functions contain references to the data in the source movie. Because the new movies contain references and not data, they are small and easily moved to and from the scrap. If you delete the movie that contains the data, the data references in the new movies are no longer valid and the new movies cannot be played. Therefore, before you delete the original movie, you should call the `FlattenMovie` function (described on page 2-105) for each of the new movies. This function copies the data into each of the new movies, eliminating the data references.

Note that the Movie Toolbox does not always copy empty tracks from the source movie to the movies that are created by these functions. Specifically, the Movie Toolbox preserves the empty tracks until you paste or add the selection into the destination movie. At that time, the Movie Toolbox removes the empty tracks from the selection. In addition, if a track in the source movie has trailing empty space, the Movie Toolbox removes that empty space from the track when it is copied into the new movie. Therefore, if you want to add a segment beyond the end of a movie, you insert the space when you insert the new segment using the `InsertMovieSegment` function (described on page 2-257).

The Movie Toolbox allows you to paste different data types into a movie. For example, QuickDraw pictures and standard sound data can be pasted directly into a movie. If you are using the movie controller component, you do not need to use these functions to paste different data types into a movie. (For details on the movie controller component, see *Inside Macintosh: QuickTime Components*.) If you are calling the Movie Toolbox directly to do editing, you should use the functions described in this section.

To get and change a movie’s current selection, your application can call the `GetMovieSelection` and `SetMovieSelection` functions.

Your application can work with a movie’s current selection by calling the `CutMovieSelection`, `CopyMovieSelection`, `PasteMovieSelection`, `ClearMovieSelection`, and `AddMovieSelection` functions.

The `PutMovieOnScrap` and `NewMovieFromScrap` functions enable your application to work with movies that are on the scrap.

Movie Toolbox

The `IsScrapMovie` function examines the system scrap to determine whether it can translate any of the data into a movie. The `PasteHandleIntoMovie` takes the contents of a specified handle, together with its type, and pastes it into a movie. `PutMovieIntoTypedHandle` takes a movie (or a single track from within a movie) and converts it into a handle.

PutMovieOnScrap

The `PutMovieOnScrap` function allows your application to place a movie onto the scrap.

```
pascal OSErr PutMovieOnScrap (Movie theMovie,
                               long movieScrapFlags);
```

theMovie Specifies the movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie`, `NewMovieFromFile`, and `NewMovieFromHandle` (described on page 2-92, page 2-88, and page 2-90, respectively).

movieScrapFlags Flags that control the operation. The following flags are available (set unused flags to 0):

`movieScrapDontZeroScrap`

Controls whether the Movie Toolbox clears the scrap before putting the movie on the scrap. If you set this flag to 1, the Movie Toolbox does not clear the scrap before placing your movie onto this scrap, thus adding your movie to the previous contents of the scrap. If you set this flag to 0, the function clears the scrap, then places your movie on the scrap.

`movieScrapOnlyPutMovie`

Controls whether the Movie Toolbox places other items on the scrap along with your movie. If you set this flag to 1, the Movie Toolbox only places your movie on the scrap. If you set this flag to 0, the Movie Toolbox places an image from the current movie time (including but not limited to a PICT) on the scrap along with your movie. The picture is intended for use by applications that cannot work with movies.

ERROR CODES

`invalidMovie` -2010 This movie is corrupted or invalid

Image Compression Manager errors

Memory Manager errors

NewMovieFromScrap

The `NewMovieFromScrap` function allows your application to create a movie from the contents of the scrap, if this is possible. If there is no movie data on the scrap, the Movie Toolbox does not create a new movie.

```
pascal Movie NewMovieFromScrap (long newMovieFlags);
```

`newMovieFlags`

Controls the operation of the `NewMovieFromScrap` function. The following flags are available (set unused flags to 0):

`newMovieActive`

Controls whether the new movie is active. Set this flag to 1 to make the new movie active. A movie that does not have any tracks can still be active. When the Movie Toolbox tries to play the movie, no images are displayed, because there is no movie data. Unless you set this flag, you should call the `SetMovieActive` function (described on page 2-145) to play a movie.

`newMovieDontResolveDataRefs`

Controls how completely the Movie Toolbox resolves data references in the movie resource. If you set this flag to 0, the toolbox tries to completely resolve all data references in the resource. This may involve searching for files on remote volumes. If you set this flag to 1, the Movie Toolbox only looks in the specified file.

If the Movie Toolbox cannot completely resolve all the data references, it still returns a valid movie identifier. In this case, the Movie Toolbox also sets the current error value to `couldNotResolveDataRef`.

`newMovieDontAskUnresolvedDataRefs`

Controls whether the Movie Toolbox asks the user to locate files. If you set this flag to 0, the Movie Toolbox asks the user to locate files that it cannot find on available volumes. If the Movie Toolbox cannot locate a file even with the user's help, the function returns a valid movie identifier and sets the current error value to `couldNotResolveDataRef`.

`newMovieDontAutoAlternate`

Controls whether the Movie Toolbox automatically selects enabled tracks from alternate track groups. If you set this flag to 1, the Movie Toolbox does not automatically select tracks for the movie—you must enable tracks yourself.

DESCRIPTION

The `NewMovieFromScrap` function returns the new movie's identifier. If the function cannot load the movie, the returned identifier is set to `nil`.

Movie Toolbox

ERROR CODES

<code>couldNotResolveDataRef</code>	-2000	Cannot use this data reference
<code>cantFindHandler</code>	-2003	Cannot locate a handler
<code>cantOpenHandler</code>	-2004	Cannot open a handler
<code>invalidMedia</code>	-2008	This media is corrupted or invalid

File Manager errors

Memory Manager errors

SetMovieSelection

The `SetMovieSelection` function sets a movie's current selection.

```
pascal void SetMovieSelection (Movie theMovie,
                               TimeValue selectionTime,
                               TimeValue selectionDuration);
```

theMovie Specifies the movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie`, `NewMovieFromFile`, and `NewMovieFromHandle` (described on page 2-92, page 2-88, and page 2-90, respectively).

selectionTime Contains a time value specifying the starting point of the current selection.

selectionDuration Contains a time value that specifies the duration of the current selection.

DESCRIPTION

If you set the `selectionDuration` parameter to a value greater than the movie's duration, `SetMovieSelection` automatically adjusts the duration of the selection to correspond to the difference between the value specified in the `selectionTime` parameter and the end of the movie.

ERROR CODES

<code>invalidMovie</code>	-2010	This movie is corrupted or invalid
<code>invalidTime</code>	-2015	This time value is invalid

SEE ALSO

You can use the `GetMovieSelection` function, described in the next section, to obtain information about a movie's current selection.

GetMovieSelection

The `GetMovieSelection` function returns information about a movie's current selection.

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

theMovie Specifies the movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie`, `NewMovieFromFile`, and `NewMovieFromHandle` (described on page 2-92, page 2-88, and page 2-90, respectively).

selectionTime Contains a pointer to a time value. The `GetMovieSelection` function places the starting time of the current selection into the field referred to by this parameter. Set this parameter to `nil` if you do not want this information.

selectionDuration Contains a pointer to a time value. The `GetMovieSelection` function places the duration of the current selection into the field referred to by this parameter. Set this parameter to `nil` if you do not want this information.

ERROR CODES

`invalidMovie` -2010 This movie is corrupted or invalid

SEE ALSO

Your application can set the current selection by calling the `SetMovieSelection` function, which is described in the previous section.

CutMovieSelection

The `CutMovieSelection` function creates a new movie that contains the original movie's current selection. This function then removes the current selection from the original 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.

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

Movie Toolbox

`theMovie` Specifies the movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie`, `NewMovieFromFile`, and `NewMovieFromHandle` (described on page 2-92, page 2-88, and page 2-90, respectively).

DESCRIPTION

The `CutMovieSelection` function returns a movie identifier. If the function could not create the new movie, it sets this returned identifier to `nil`.

Your application must dispose of the new movie once you are done with it. You can use the `DisposeMovie` function (described on page 2-96) to dispose of the new movie.

If you have assigned a progress function to the source movie, the Movie Toolbox calls that progress function during long cut operations. (For details on progress functions, see “Progress Functions” beginning on page 2-354.)

ERROR CODES

<code>invalidMovie</code>	-2010	This movie is corrupted or invalid
<code>progressProcAborted</code>	-2019	Your progress function returned an error
Memory Manager errors		

CopyMovieSelection

The `CopyMovieSelection` function creates a new movie that contains the original movie’s current selection. This function does not change the original movie or the current selection.

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

`theMovie` Specifies the movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie`, `NewMovieFromFile`, and `NewMovieFromHandle` (described on page 2-92, page 2-88, and page 2-90, respectively).

DESCRIPTION

The `CopyMovieSelection` function returns a movie identifier. If the function could not create the new movie, it sets this returned identifier to `nil`.

Your application must dispose of the new movie once you are done with it. You can use the `DisposeMovie` function (described on page 2-96) to dispose of the new movie.

If you have assigned a progress function to the source movie, the Movie Toolbox calls that progress function during long copy operations. (For details on progress functions, see “Progress Functions” beginning on page 2-354.)

ERROR CODES

<code>invalidMovie</code>	-2010	This movie is corrupted or invalid
<code>progressProcAborted</code>	-2019	Your progress function returned an error
Memory Manager errors		

PasteMovieSelection

The `PasteMovieSelection` function places the tracks from one movie into another movie.

```
pascal void PasteMovieSelection (Movie theMovie, Movie src);
```

theMovie Specifies the destination movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie`, `NewMovieFromFile`, and `NewMovieFromHandle` (described on page 2-92, page 2-88, and page 2-90, respectively).

src Specifies the source movie for this operation. The `PasteMovieSelection` function places the tracks from this movie in the destination movie.

DESCRIPTION

All of the tracks from the source movie are placed in the destination movie. If the duration of the destination movie's current selection is 0, the source movie is inserted at the starting time of the current selection. If the current selection duration is nonzero, the function clears the current selection and then inserts the tracks from the source movie. After the paste operation, the current selection time is unchanged, and the selection duration is set to the source movie's duration.

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. The toolbox considers the following characteristics when searching for an appropriate track:

- track spatial dimensions
- track matrix
- track clipping region
- track matte
- alternate group affiliation
- media time scale
- media type
- media language
- data reference (that is, the two tracks must refer to the same file)

Movie Toolbox

If the Movie Toolbox cannot find an appropriate track in the destination movie, it creates a track with the proper characteristics.

The Movie Toolbox removes any empty tracks from the destination movie after the paste operation.

If you have assigned a progress function to the destination movie, the Movie Toolbox calls that progress function during long paste operations. (For details on progress functions, see “Progress Functions” beginning on page 2-354.)

SPECIAL CONSIDERATIONS

The entire source movie is used regardless of the selection in the source movie.

ERROR CODES

<code>invalidMovie</code>	-2010	This movie is corrupted or invalid
<code>progressProcAborted</code>	-2019	Your progress function returned an error
Memory Manager errors		

SEE ALSO

If you want to insert only a part of the source movie, use the `InsertMovieSegment` function, which is described on page 2-257.

AddMovieSelection

The `AddMovieSelection` function adds one or more tracks to a movie. 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.

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

<code>theMovie</code>	Specifies the destination movie for this operation. Your application obtains this movie identifier from such functions as <code>NewMovie</code> , <code>NewMovieFromFile</code> , and <code>NewMovieFromHandle</code> (described on page 2-92, page 2-88, and page 2-90, respectively).
<code>src</code>	Specifies the source movie for this operation. The <code>AddMovieSelection</code> function 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.

DESCRIPTION

The `AddMovieSelection` function is similar to `PasteMovieSelection`, which is described in the previous section. However, the `PasteMovieSelection` function inserts empty space into a movie's existing tracks and then adds the new track data. The `AddMovieSelection` function does not insert empty space into the existing tracks. This function simply adds the tracks in parallel from the source movie to the destination movie. This can be useful for adding a track to an existing movie, such as adding sound to a silent 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. (For details, see "Progress Functions" beginning on page 2-354.)

The entire source movie is used regardless of the selection in the source movie.

ERROR CODES

<code>invalidMovie</code>	-2010	This movie is corrupted or invalid
<code>progressProcAborted</code>	-2019	Your progress function returned an error
Memory Manager errors		

ClearMovieSelection

The `ClearMovieSelection` function removes the segment of the movie that is defined by the current selection.

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

`theMovie` Specifies the movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie`, `NewMovieFromFile`, and `NewMovieFromHandle` (described on page 2-92, page 2-88, and page 2-90, respectively).

DESCRIPTION

After removing the segment, the Movie Toolbox sets the duration of the movie's current selection to 0 and the selection time remains unchanged. This function removes empty tracks from the resulting movie.

ERROR CODES

<code>invalidMovie</code>	-2010	This movie is corrupted or invalid
---------------------------	-------	------------------------------------

IsScrapMovie

The `IsScrapMovie` function looks on the system scrap to find out if it can translate any of the data into a movie.

```
pascal Component IsScrapMovie (Track targetTrack);
```

`targetTrack`

Specifies the location of the potential target movie for the data on the system scrap.

DESCRIPTION

If `IsScrapMovie` finds an appropriate type, it returns a movie import component that can translate the scrap. Otherwise, it returns 0. For details on movie import components, see *Inside Macintosh: QuickTime Components*.

PasteHandleIntoMovie

The `PasteHandleIntoMovie` function takes the contents of a specified handle, together with its type, and pastes it into a specified movie.

```
pascal OSErr PasteHandleIntoMovie (Handle h, OSType handleType,
                                   Movie theMovie, long flags,
                                   ComponentInstance userComp);
```

`h` Specifies the handle to be pasted into the movie indicated by the `handleType` parameter.

`handleType`

Indicates the data type of the handle specified in the `h` parameter.

`theMovie`

Specifies the destination movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie`, `NewMovieFromFile`, and `NewMovieFromHandle` (described on page 2-92, page 2-88, and page 2-90, respectively).

`flags`

Specifies a constant that further refines conditions of the paste operation.

`pasteInParallel`

Changes the function so that it takes the contents of the specified handle along with its type and adds (rather than inserts) it to the specified movie in an operation analogous to that of the `AddMovieSelection` function. This operation does not affect the duration of existing tracks. It does not necessarily create a new track; rather, it uses a piece of an existing track, if possible.

Movie Toolbox

userComp Specifies 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, it will be used by `PasteHandleIntoMovie`. 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.

DESCRIPTION

If the handle is set to 0, `PasteHandleIntoMovie` searches the scrap for a field of the type `handleType`. If both the `h` parameter and the `handleType` parameter are `nil`, `PasteHandleIntoMovie` uses the first available data from the scrap.

If you are just pasting in data from the scrap, it is best to allow `PasteHandleIntoMovie` 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').

`PasteHandleIntoMovie` pastes into the current selection according to the following rules:

- If the selection is empty (for example, `duration = 0`), `PasteHandleIntoMovie` 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.

PutMovieIntoTypedHandle

The `PutMovieIntoTypedHandle` function takes a movie (or a single track from within that movie) and converts it into a handle of a specified type.

```
pascal OSErr PutMovieIntoTypedHandle (Movie theMovie,
                                       Track targetTrack,
                                       OSType handleType,
                                       Handle publicMovie,
                                       TimeValue start,
                                       TimeValue dur,
                                       long flags,
                                       ComponentInstance userComp);
```


Movie Toolbox

<code>theMovie</code>	Specifies the movie to convert.
<code>targetTrack</code>	Specifies the track to convert.
<code>handleType</code>	Indicates the type of the new data.
<code>publicMovie</code>	Contains the actual handle in which to place the new data.
<code>start</code>	Specifies the start time of the segment of the movie or track to be converted.
<code>dur</code>	Specifies the duration of the segment of the movie or track to be converted.
<code>flags</code>	Indicates condition of the conversion. Set this parameter to 0.
<code>userComp</code>	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 <code>PutMovieIntoTypedHandle</code> . 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. For details on movie export components, see <i>Inside Macintosh: QuickTime Components</i> .

Undo for Movies

The Movie Toolbox provides functions that allow you to capture and restore the edit state of a movie. An **edit state** contains information that completely defines a movie's content at the time you create the edit state. It is, in essence, a checkpoint in the edit session. You can manage a movie's edit states in order to implement an undo capability for editing movies. For example, you can capture a movie's edit state before performing an editing operation, such as a cut, and later restore the old state. You can have several movie edit states obtained at different times during an editing session, and restore to any one of them at any time. In this manner, you can provide a multilevel undo capability. This section describes the Movie Toolbox functions that work with edit states.

Note that a movie's edit state does not save everything about a movie. Most important, the edit state does not contain information about the movie's spatial characteristics. For example, the edit state does not store the current boundary rectangle or clipping region. Consequently, edit states are best suited to supporting undo operations involving movie content, including track creation and removal. You can use other Movie Toolbox functions to support undo operations for movie characteristics. See "Functions That Modify Movie Properties" beginning on page 2-157 to learn more about these functions.

You can use the `NewMovieEditState` function to capture a movie's edit state. Use the `UseMovieEditState` to restore the movie to its condition according to a previous edit state. Your application must dispose of an edit state by calling the `DisposeMovieEditState` function. You must dispose of a movie's edit states before you dispose of the movie.

NewMovieEditState

You can create an edit state by calling the `NewMovieEditState` function. This function creates an edit state that 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.

Note

You must dispose of a movie's edit states *before* you dispose of the movie itself. Use the `DisposeMovieEditState` function (described on page 2-256) to dispose of an edit state. ♦

```
pascal MovieEditState NewMovieEditState (Movie theMovie);
```

theMovie Specifies the movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie`, `NewMovieFromFile`, and `NewMovieFromHandle` (described on page 2-92, page 2-88, and page 2-90, respectively).

DESCRIPTION

The `NewMovieEditState` function returns a movie edit state identifier. You can use this identifier with other Movie Toolbox edit state functions, such as `UseMovieEditState` (described in the next section). If this function could not create the edit state, it sets this returned identifier to `nil`.

ERROR CODES

`invalidMovie` -2010 This movie is corrupted or invalid
Memory Manager errors

UseMovieEditState

Your application can use the `UseMovieEditState` function to return a movie to its condition according to an edit state you created previously.

```
pascal OSErr UseMovieEditState (Movie theMovie,  
                                MovieEditState toState);
```

theMovie Specifies the movie for this operation. Your application obtains this movie identifier from such functions as `NewMovie`, `NewMovieFromFile`, and `NewMovieFromHandle` (described on page 2-92, page 2-88, and page 2-90, respectively).

Movie Toolbox

`toState` Specifies the edit state for this operation. Your application obtains this edit state identifier when you create the edit state by calling the `NewMovieEditState` function (described in the previous section).

DESCRIPTION

The `UseMovieEditState` function uses the information stored in the edit state to update the movie's contents. This may change the contents of some of the movie's tracks, or it may even add tracks to the movie or remove tracks from the movie. Consequently, the movie's time and spatial characteristics, especially the duration, may change as a result of restoring the saved edit state. Your application creates an edit state by calling the `NewMovieEditState` function, which is described in the previous section.

ERROR CODES

<code>invalidMovie</code>	-2010	This movie is corrupted or invalid
<code>invalidEditState</code>	-2023	This edit state is invalid
<code>nonMatchingEditState</code>	-2024	This edit state is not valid for this movie
<code>staleEditState</code>	-2025	Movie or track has been disposed

DisposeMovieEditState

The `DisposeMovieEditState` function disposes of an edit state. Your application must dispose of any edit states you create.

Note

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

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

`state` Specifies the edit state for this operation. Your application obtains this edit state identifier when you create the edit state by calling the `NewMovieEditState` function.

ERROR CODES

<code>invalidEditState</code>	-2023	This edit state is invalid
<code>staleEditState</code>	-2025	Movie or track has been disposed

SEE ALSO

You create an edit state by calling the `NewMovieEditState` function, which is discussed on page 2-255.

Low-Level Movie-Editing Functions

The Movie Toolbox provides a number of functions that allow your application to perform low-level editing operations on movies. These functions work with movie segments—pieces of a movie that are defined by a starting time and duration—and therefore give you a great deal of control over the editing process. These functions never copy the movie data; rather, they work with references to the movie’s data. “Editing Movies,” which begins on page 2-243, discusses the Movie Toolbox functions that allow you to edit movies by working with the current selection.

You can use the `CopyMovieSettings` function to copy certain important settings from one movie to another.

You can use the `InsertMovieSegment` function to copy a segment from one movie to another. Use the `InsertMovieEmptySegment` function to insert an empty segment into a movie.

Your application can delete a segment from a movie by calling the `DeleteMovieSegment` function.

You can change a segment’s duration by calling the `ScaleMovieSegment` function. This function stretches or shrinks the segment to accommodate a specified duration.

InsertMovieSegment

The `InsertMovieSegment` function copies part of one movie to another. You specify the starting time and duration of the source segment and the time in the destination movie at which to place the information.

```
pascal OSErr InsertMovieSegment (Movie srcMovie, Movie dstMovie,
                                TimeValue srcIn,
                                TimeValue srcDuration,
                                TimeValue dstIn);
```

- | | |
|-----------------------|--|
| <code>srcMovie</code> | Specifies the source movie for this operation. Your application obtains this movie identifier from such functions as <code>NewMovie</code> , <code>NewMovieFromFile</code> , and <code>NewMovieFromHandle</code> (described on page 2-92, page 2-88, and page 2-90, respectively). The <code>InsertMovieSegment</code> function obtains the movie segment from the source movie specified in this parameter. |
| <code>dstMovie</code> | Specifies the destination movie for this operation. The <code>InsertMovieSegment</code> function places a copy of the segment, which is obtained from the source movie, into this destination movie. The <code>dstIn</code> parameter specifies where the segment is inserted. |
| <code>srcIn</code> | Specifies the start of the segment in the source movie. The <code>srcDuration</code> parameter specifies the segment’s duration. This time value must be expressed in the source movie’s time scale. |

Movie Toolbox

`srcDuration`

Specifies the duration of the segment in the source movie. This time value must be expressed in the source movie's time scale.

`dstIn`

Contains a time value specifying where the segment is to be inserted. This time value must be expressed in the destination movie's time scale.

DESCRIPTION

The `InsertMovieSegment` function does not change the source movie. However, the duration of the destination movie is extended to accommodate the inserted segment. You can use this function to add a segment beyond the end of the destination movie—the Movie Toolbox inserts empty space as appropriate.

You can use the `InsertMovieSegment` function to copy data within a single movie. If you are not copying data from one location in a movie to a different point in the same movie, the function may create new tracks, as appropriate.

Whenever possible, the Movie Toolbox uses existing tracks to store the data to be inserted. 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. The toolbox considers the following characteristics when searching for an appropriate track:

- track spatial dimensions
- track matrix
- track clipping region
- track matte
- alternate group affiliation
- media time scale
- media type
- media language
- 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.

If you have assigned a progress function to the destination movie, the Movie Toolbox calls that progress function during long copy operations. For details on application-defined progress functions, see “Progress Functions” beginning on page 2-354.

ERROR CODES

<code>invalidMovie</code>	-2010	This movie is corrupted or invalid
<code>invalidDuration</code>	-2014	This duration value is invalid
<code>invalidTime</code>	-2015	This time value is invalid
<code>progressProcAborted</code>	-2019	Your progress function returned an error
Memory Manager errors		

InsertEmptyMovieSegment

The `InsertEmptyMovieSegment` function adds an empty segment to a movie. 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.

```
pascal OSErr InsertEmptyMovieSegment (Movie dstMovie,
                                       TimeValue dstIn,
                                       TimeValue dstDuration);
```

<code>dstMovie</code>	Specifies the movie for this operation. Your application obtains this movie identifier from such functions as <code>NewMovie</code> , <code>NewMovieFromFile</code> , and <code>NewMovieFromHandle</code> (described on page 2-92, page 2-88, and page 2-90, respectively).
<code>dstIn</code>	Contains a time value specifying where the segment is to be inserted. This time value must be expressed in the movie's time scale.
<code>dstDuration</code>	Contains a time value that specifies the duration of the segment to be added.

DESCRIPTION

The `InsertEmptyMovieSegment` function then inserts the appropriate amount of empty time into each of the movie's tracks. The exact meaning of the term *empty time* depends upon the type of track. For example, empty time in a sound track is silent.

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 the `InsertMovieSegment` function, which is described in the previous section.

ERROR CODES

<code>invalidMovie</code>	-2010	This movie is corrupted or invalid
<code>invalidDuration</code>	-2014	This duration value is invalid
<code>invalidTime</code>	-2015	This time value is invalid
Memory Manager errors		

DeleteMovieSegment

The `DeleteMovieSegment` function removes a specified segment from a movie. You identify the segment to remove by specifying its starting time and duration.

```
pascal OSErr DeleteMovieSegment (Movie theMovie, TimeValue in,
                                TimeValue duration);
```

<code>theMovie</code>	Specifies the movie for this operation. Your application obtains this movie identifier from such functions as <code>NewMovie</code> , <code>NewMovieFromFile</code> , and <code>NewMovieFromHandle</code> (described on page 2-92, page 2-88, and page 2-90, respectively).
<code>in</code>	Contains a time value specifying the starting point of the segment to be deleted.
<code>duration</code>	Contains a time value that specifies the duration of the segment to be deleted.

ERROR CODES

<code>invalidMovie</code>	-2010	This movie is corrupted or invalid
<code>invalidDuration</code>	-2014	This duration value is invalid
<code>invalidTime</code>	-2015	This time value is invalid

ScaleMovieSegment

The `ScaleMovieSegment` function changes the duration of a segment of a movie. The Movie Toolbox scales the segment to accommodate the new duration.

```
pascal OSErr ScaleMovieSegment (Movie theMovie, TimeValue in,
                                TimeValue oldDuration,
                                TimeValue newDuration);
```

<code>theMovie</code>	Specifies the movie for this operation. Your application obtains this movie identifier from such functions as <code>NewMovie</code> , <code>NewMovieFromFile</code> , and <code>NewMovieFromHandle</code> (described on page 2-92, page 2-88, and page 2-90, respectively).
<code>in</code>	Specifies the start of the segment. The <code>oldDuration</code> parameter specifies the segment's duration. This time value must be expressed in the movie's time scale.
<code>oldDuration</code>	Specifies the duration of the segment in the source movie. This time value must be expressed in the movie's time scale.

Movie Toolbox

`newDuration`
Specifies 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.

ERROR CODES

<code>invalidMovie</code>	–2010	This movie is corrupted or invalid
<code>invalidDuration</code>	–2014	This duration value is invalid
<code>invalidTime</code>	–2015	This time value is invalid
Memory Manager errors		

CopyMovieSettings

The `CopyMovieSettings` function copies many settings from one movie to another, overwriting the destination settings in the process.

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

<code>srcMovie</code>	Specifies the source movie for this operation. Your application obtains this movie identifier from such functions as <code>NewMovie</code> , <code>NewMovieFromFile</code> , and <code>NewMovieFromHandle</code> (described on page 2-92, page 2-88, and page 2-90, respectively).
<code>dstMovie</code>	Specifies the destination movie for this operation. The <code>CopyMovieSettings</code> function uses the settings from the source movie, which is specified by the <code>srcMovie</code> parameter, to replace the current settings of this movie.

DESCRIPTION

The `CopyMovieSettings` function copies the

- preferred rate and volume
- source clipping region
- matrix information
- user data

If you want to work with specific characteristics, you can use the Movie Toolbox functions that allow you to manipulate movie settings individually. These functions are described in “Functions That Modify Movie Properties” beginning on page 2-157.

This function does not copy the movie’s contents. To work with movie contents, you should use the segment editing functions described in “Low-Level Movie-Editing Functions” beginning on page 2-257.

ERROR CODES

`invalidMovie` -2010 This movie is corrupted or invalid
 Memory Manager errors

Editing Tracks

The Movie Toolbox provides a number of functions that allow your application to perform editing operations on tracks. These functions work with track segments—pieces of a track that are defined by a starting time and duration—and therefore give you a great deal of control over the editing process. These functions are similar to the low-level editing functions for movies that were described earlier in this chapter. However, these functions may copy movie data, if required by the operation.

When you edit a track you may change the duration of the movie that contains that track.

The `CopyTrackSettings` function lets you copy certain important settings from one track to another.

You can use the `InsertTrackSegment` function to copy a segment from one track to another. The `InsertTrackEmptySegment` function allows you to insert an empty segment into a track.

You can use the `InsertMediaIntoTrack` function to insert a media into a track.

Your application can delete a segment from a track by calling the `DeleteTrackSegment` function.

You can change a segment's duration by calling the `ScaleTrackSegment` function. This function stretches or shrinks the segment to accommodate a specified duration.

You can use the `GetTrackEditRate` function to determine the rate of the track edit of a specified track at an indicated time.

InsertTrackSegment

The `InsertTrackSegment` function copies part of one track to another. You specify the starting time and duration of the source segment and the time in the destination track at which to place the information.

```
pascal OSErr InsertTrackSegment (Track srcTrack, Track dstTrack,
                                TimeValue srcIn,
                                TimeValue srcDuration,
                                TimeValue dstIn);
```

`srcTrack` Specifies the source track for this operation. Your application obtains this track identifier from such Movie Toolbox functions as `NewMovieTrack` and `GetMovieTrack` (described on page 2-151 and page 2-204, respectively).

Movie Toolbox

<code>dstTrack</code>	Specifies the destination track for this operation. The <code>InsertTrackSegment</code> function places a copy of the segment, which is obtained from the source track, into this destination track. The <code>in</code> parameter specifies where the segment is inserted.
<code>srcIn</code>	Specifies the start of the segment in the source track. The <code>srcDuration</code> parameter specifies the segment's duration. This time value must be expressed in the time scale of the movie that contains the source track.
<code>srcDuration</code>	Specifies 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.
<code>dstIn</code>	Contains 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.

DESCRIPTION

The `InsertTrackSegment` function does not change the source track. However, the duration of the destination track is extended to accommodate the inserted segment. This may also change the duration of the movie that contains the destination track.

You can use this function to copy data within a single track. If you are not copying data from one location in a track to a different point in the same track, 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.

In addition, if the source and destination tracks are associated with different media data files, this function copies samples from the source to the destination using the `AddMediaSample` function. Therefore, the Movie Toolbox must be able to write to the destination media. In this case, your application must call the `BeginMediaEdits` function before calling `InsertTrackSegment`. At the end of the editing session, your application must call the `EndMediaEdits` function. See “Adding Samples to Media Structures” beginning on page 2-271 for more information about these functions.

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.

ERROR CODES

<code>invalidTrack</code>	-2009	This track is corrupted or invalid
<code>mediaTypesDontMatch</code>	-2018	These media structures don't match
<code>progressProcAborted</code>	-2019	Your progress function returned an error
File Manager errors		

InsertEmptyTrackSegment

The `InsertEmptyTrackSegment` function adds an empty segment to a track. 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 silent.

```
pascal OSErr InsertEmptyTrackSegment (Track dstTrack,
                                       TimeValue dstIn,
                                       TimeValue dstDuration);
```

dstTrack Specifies the track for this operation. Your application obtains this track identifier from such Movie Toolbox functions as `NewMovieTrack` and `GetMovieTrack` (described on page 2-151 and page 2-204, respectively).

dstIn Contains 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 Contains 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.

DESCRIPTION

Note that you cannot add empty space to the end of a movie or to the end of a track. 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`.

ERROR CODES

<code>invalidTrack</code>	-2009	This track is corrupted or invalid
<code>invalidDuration</code>	-2014	This duration value is invalid
<code>invalidTime</code>	-2015	This time value is invalid

Memory Manager errors

InsertMediaIntoTrack

The `InsertMediaIntoTrack` function inserts a reference to a media segment into a track. 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.

```
pascal OSErr InsertMediaIntoTrack (Track theTrack,
                                   TimeValue trackStart,
                                   TimeValue mediaTime,
                                   TimeValue mediaDuration,
                                   Fixed mediaRate);
```

- `theTrack` Specifies the track for this operation. Your application obtains this track identifier from such Movie Toolbox functions as `NewMovieTrack` and `GetMovieTrack` (described on page 2-151 and page 2-204, respectively).
- `trackStart` Contains 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` Contains 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` Contains a time value specifying the duration of the media's segment. This time value must be expressed in the media's time scale.
- `mediaRate` Specifies the media's rate. A value of 1.0 indicates the media's natural playback rate. This value should be a positive, nonzero rate.

DESCRIPTION

The `InsertMediaIntoTrack` function 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.

You use this function after you have added samples to a media using the functions described in "Adding Samples to Media Structures" beginning on page 2-271. If you play the track before you call this function, the track does not contain the new media data.

ERROR CODES

<code>invalidTrack</code>	-2009	This track is corrupted or invalid
<code>invalidDuration</code>	-2014	This duration value is invalid
<code>invalidTime</code>	-2015	This time value is invalid

DeleteTrackSegment

The `DeleteTrackSegment` function removes a specified segment from a track. You identify the segment to remove by specifying its starting time and duration.

```
pascal OSErr DeleteTrackSegment (Track theTrack, TimeValue in,
                                TimeValue duration);
```

<code>theTrack</code>	Specifies the track for this operation. Your application obtains this track identifier from such Movie Toolbox functions as <code>NewMovieTrack</code> and <code>GetMovieTrack</code> (described on page 2-151 and page 2-204, respectively).
<code>in</code>	Contains 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.
<code>duration</code>	Contains 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.

ERROR CODES

<code>invalidTrack</code>	-2009	This track is corrupted or invalid
<code>invalidDuration</code>	-2014	This duration value is invalid
<code>invalidTime</code>	-2015	This time value is invalid

SEE ALSO

To dispose of a track, call the `DisposeMovieTrack` function, described on page 2-152.

ScaleTrackSegment

The `ScaleTrackSegment` function changes the duration of a segment of a track. This may change the duration of the movie that contains the track. However, this function does not cause the Movie Toolbox to add data to or remove data from the movie.

```
pascal OSErr ScaleTrackSegment (Track theTrack, TimeValue in,
                                TimeValue oldDuration,
                                TimeValue newDuration);
```

<code>theTrack</code>	Specifies the track for this operation. Your application obtains this track identifier from such Movie Toolbox functions as <code>NewMovieTrack</code> and <code>GetMovieTrack</code> (described on page 2-151 and page 2-204, respectively).
<code>in</code>	Specifies the start of the segment. The <code>oldDuration</code> parameter specifies the segment's duration. This time value must be expressed in the time scale of the movie that contains the track.

Movie Toolbox

`oldDuration`
Specifies the duration of the segment. This time value must be expressed in the time scale of the movie that contains the track.

`newDuration`
Specifies 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.

ERROR CODES

<code>invalidTrack</code>	-2009	This track is corrupted or invalid
<code>invalidDuration</code>	-2014	This duration value is invalid
<code>invalidTime</code>	-2015	This time value is invalid
Memory Manager errors		

CopyTrackSettings

The `CopyTrackSettings` function copies many settings from one track to another, overwriting the destination settings.

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

`srcTrack` Specifies the source track for this operation. Your application obtains this track identifier from such Movie Toolbox functions as `NewMovieTrack` and `GetMovieTrack` (described on page 2-151 and page 2-204, respectively).

`dstTrack` Specifies 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.

DESCRIPTION

The `CopyTrackSettings` function copies the

- matrix information
- track volume
- clipping region
- user data
- matte information
- media language, quality, and user data
- other media-specific settings (such as sound balance and video graphics mode)

This function does not copy any alternate group information pertaining to the track.

Movie Toolbox

If you want to work with specific characteristics, you can use the Movie Toolbox functions that allow you to manipulate track settings individually. These functions are described in “Functions That Modify Movie Properties,” which begins on page 2-157.

This function does not copy the track’s contents. To work with track contents, you should use the segment-editing functions described in “Editing Tracks” beginning on page 2-262.

ERROR CODES

invalidTrack -2009 This track is corrupted or invalid
Memory Manager errors

GetTrackEditRate

The `GetTrackEditRate` function returns the rate of the track edit of a specified track at an indicated time.

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

`theTrack` Specifies 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.

DESCRIPTION

If an invalid time or track is passed, the returned value is 0.0. The track edit rate is typically 1.0, unless either the `ScaleMovieSegment` or `ScaleTrackSegment` function has been called. (For more on the `ScaleMovieSegment` and `ScaleTrackSegment` functions, see page 2-260 and page 2-266, respectively.)

The `GetTrackEditRate` function is relevant if you are stepping through track edits directly in your application or if you are a client of the **base media handler**. (See *Inside Macintosh: QuickTime Components* for details on media handlers.)

Undo for Tracks

The Movie Toolbox provides functions that allow you to capture and restore the edit state of a track. As with the functions that manipulate a movie’s edit state, you can manage a track’s edit states in order to implement an undo capability for track editing. For example, you can capture a track’s edit state before performing an editing operation, such as a cut, and later restore the old state. You can have several track edit states obtained at different times during an editing session, and you can restore to any one of

Movie Toolbox

them at any time. In this manner, you can provide a multilevel undo capability. This section describes the Movie Toolbox functions that work with track edit states.

Note that a track's edit state does not save everything about the track. Most important, the edit state does not contain information about track spatial characteristics. For example, the edit state does not store the current clipping region. Consequently, edit states are best suited to supporting undo operations involving track content. You can use other Movie Toolbox functions to support undo operations for track characteristics. See "Functions That Modify Movie Properties," which begins on page 2-157, to learn more about these functions.

You can use the `NewTrackEditState` function to capture a track's edit state. Use the `UseTrackEditState` function to restore the track to its condition according to a previous edit state. Your application can dispose of an edit state by calling the `DisposeTrackEditState` function.

NewTrackEditState

You can create an edit state by calling the `NewTrackEditState` function. This function creates an edit state that 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.

Note

You must dispose of a movie's track edit states *before* disposing of the track or of the movie that contains the track. Use the `DisposeTrackEditState` function, which is described on page 2-270, to dispose of an edit state. ♦

```
pascal TrackEditState NewTrackEditState (Track theTrack);
```

theTrack Specifies the track for this operation. Your application obtains this track identifier from such Movie Toolbox functions as `NewMovieTrack` and `GetMovieTrack` (described on page 2-151 and page 2-204, respectively).

DESCRIPTION

The `NewTrackEditState` function returns a track edit state identifier. You can use this identifier with other Movie Toolbox edit state functions, such as `UseTrackEditState` (described in the next section). If this function could not create the edit state, it sets this returned identifier to `nil`.

ERROR CODES

<code>invalidTrack</code>	-2009	This track is corrupted or invalid
Memory Manager errors		

UseTrackEditState

Your application can use the `UseTrackEditState` function to return a track to its condition according to an edit state you created previously.

```
pascal OSErr UseTrackEditState (Track theTrack,
                                TrackEditState state);
```

theTrack Specifies the track for this operation. Your application obtains this track identifier from such Movie Toolbox functions as `NewMovieTrack` and `GetMovieTrack` (described on page 2-151 and page 2-204, respectively).

state Specifies the edit state for this operation. Your application obtains this edit state identifier when you create the edit state by calling the `NewTrackEditState` function, which is described in the previous section.

DESCRIPTION

The `UseTrackEditState` function uses the information stored in the edit state to update the track's contents. This may change the contents of some of the track. Consequently, the time characteristics of the movie that contains the track, especially the duration, may change as a result of restoring the saved edit state. Your application creates an edit state by calling the `NewTrackEditState` function.

SPECIAL CONSIDERATIONS

You can use the `UseTrackEditState` function only with tracks that currently belong to a movie. A track may be detached from its movie as a result of edit processing—you cannot use this function with such a track.

ERROR CODES

<code>invalidTrack</code>	-2009	This track is corrupted or invalid
<code>invalidEditState</code>	-2023	This edit state is invalid
<code>nonMatchingEditState</code>	-2024	This edit state is not valid for this movie

DisposeTrackEditState

The `DisposeTrackEditState` function disposes of a track edit state. Your application must dispose of any edit states you create. You create an edit state by calling the `NewTrackEditState` function, which is discussed on page 2-269.

Note

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

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

state Specifies the edit state for this operation. Your application obtains this edit state identifier when you create the edit state by calling the `NewTrackEditState` function (described on page 2-269).

ERROR CODES

<code>invalidTrack</code>	-2009	This track is corrupted or invalid
<code>invalidEditState</code>	-2023	This edit state is invalid
<code>staleEditState</code>	-2025	Movie or track has been disposed

Adding Samples to Media Structures

This section describes Movie Toolbox functions that directly manipulate media samples. These functions are used only by applications that create movies or add data to existing movies.

You add samples to a media by calling the `AddMediaSample` function. You can indicate that the sample to be added is not a sync sample. **Sync samples** do not rely on preceding frames for content. Some compression algorithms conserve space by eliminating duplication between consecutive frames in a sample. In image data, sync samples are referred to as *key frames*. For more information on key frames, see the chapter “Image Compression Manager” in this book.

You can obtain the data in a media sample by calling the `GetMediaSample` function. If you are going to add samples to a media, you must do so within a media-editing session. You start a media-editing session by calling the `BeginMediaEdits` function. Once you have finished adding samples to the media, you end the editing session by calling the `EndMediaEdits` function.

Once you have added samples to a media, you can work with references to those samples by calling the `AddMediaSampleReference` and `GetMediaSampleReference` functions. You do not have to be in a media-editing session to use these functions.

BeginMediaEdits

The `BeginMediaEdits` function starts a media-editing session.

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

theMedia Specifies the media for this operation. Your application obtains this media identifier from such Movie Toolbox functions as `NewTrackMedia` and `GetTrackMedia` (described on page 2-153 and page 2-206, respectively).

Movie Toolbox

DESCRIPTION

You use the `BeginMediaEdits` function to notify the Movie Toolbox that you are going to add sample data to a media. In response, the Movie Toolbox determines whether the media can be updated. For example, if the media data are stored on disk, the Movie Toolbox opens the disk file with write permissions. If the media is stored on a read-only storage medium, such as a CD-ROM disc, the Movie Toolbox does not start an editing session and returns an error.

Use the `EndMediaEdits` function, which is described in the next section, to end a media-editing session.

You must call `BeginMediaEdits` before you add samples to a media with the `AddMediaSample` function (described on page 2-273). Under some circumstances, you must start a media-editing session before calling the `InsertTrackSegment` function (described on page 2-262).

ERROR CODES

<code>invalidMedia</code>	-2008	This media is corrupted or invalid
File system errors		

EndMediaEdits

The `EndMediaEdits` function ends a media-editing session.

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

<code>theMedia</code>	Specifies the media for this operation. Your application obtains this media identifier from such Movie Toolbox functions as <code>NewTrackMedia</code> and <code>GetTrackMedia</code> (described on page 2-153 and page 2-206, respectively).
-----------------------	---

DESCRIPTION

You use the `EndMediaEdits` function to tell the Movie Toolbox that you are done adding samples to a movie data file. The Movie Toolbox then performs the appropriate processing. For example, for disk-based media, the Movie Toolbox relinquishes write-access to the disk file. You should call `EndMediaEdits` only if you successfully started a media-editing session with the `BeginMediaEdits` function, which is described in the previous section.

ERROR CODES

<code>invalidMedia</code>	-2008	This media is corrupted or invalid
---------------------------	-------	------------------------------------

AddMediaSample

The `AddMediaSample` function adds sample data and a description to a media. Your application specifies the sample and the media for the operation. The `AddMediaSample` function 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, each time a sample is added, the media duration is extended.

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

<code>theMedia</code>	Specifies the media for this operation. Your application obtains this media identifier from such Movie Toolbox functions as <code>NewTrackMedia</code> and <code>GetTrackMedia</code> (described on page 2-153 and page 2-206, respectively).
<code>dataIn</code>	Contains a handle to the sample data. The <code>AddMediaSample</code> function adds this data to the media specified by the parameter <code>theMedia</code> . You specify the number of bytes of sample data with the <code>size</code> parameter. You can use the <code>inOffset</code> parameter to specify a byte offset into the data referred to by this handle.
<code>inOffset</code>	Specifies an offset into the data referred to by the handle contained in the <code>dataIn</code> parameter. Set this parameter to 0 if there is no offset.
<code>size</code>	Specifies 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 <code>numberOfSamples</code> parameter to indicate the number of samples that are contained in the sample data.
<code>durationPerSample</code>	Specifies 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 the <code>durationPerSample</code> parameter 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.
<code>sampleDescriptionH</code>	Contains a handle to a sample description. Some media structures may require sample descriptions. There are different sample descriptions for different types of samples. For example, a media that contains compressed video requires that you supply an image description (see the chapter "Image Compression Manager" in this book for more information about image description structures). A media that contains sound requires

Movie Toolbox

that you supply a sound description structure (see “The Sound Description Structure” on page 2-79 for more information about sound description structures).

If the media does not require a sample description, set this parameter to `nil`.

`numberOfSamples`

Specifies the number of samples contained in the sample data to be added to the media.

This parameter determines the size of each sample. 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 that control the add operation. The following flag is available (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 this flag to 0 if the sample is a sync sample.

`sampleTime`

Contains 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 do not want to receive this information, set this parameter to `nil`.

DESCRIPTION

The `AddMediaSample` function updates the file or device that contains the movie data file as part of the add operation. Consequently, your application must have started a media-editing session before calling this function. You start a media-editing session with the `BeginMediaEdits` function, which is described on page 2-271. If you want to work with samples that have already been added to a movie data file, use the `AddMediaSampleReference` function, which is described in the next section.

ERROR CODES

invalidMedia -2008 This media is corrupted or invalid

File Manager errors

Memory Manager errors

AddMediaSampleReference

The `AddMediaSampleReference` function allows your application to work with samples that have already been 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 the `dataOffset` and `size` parameters.

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

theMedia Specifies the media for this operation. Your application obtains this media identifier from such Movie Toolbox functions as `NewTrackMedia` and `GetTrackMedia` (described on page 2-153 and page 2-206, respectively).

dataOffset Specifies 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 the `GetMediaSampleReference` function (described on page 2-279).

size Specifies 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 the `numberOfSamples` parameter to indicate the number of samples that are contained in the reference.

durationPerSample Specifies 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 the `durationPerSample` parameter 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.

Movie Toolbox

`sampleDescriptionH`

Contains a handle to a sample description. Some media structures may require sample descriptions. There are different sample descriptions for different types of samples. For example, a media that contains compressed video requires that you supply an image description (see the chapter “Image Compression Manager” in this book for more information about image description structures). A media that contains sound requires that you supply a sound description structure (see “The Sound Description Structure” on page 2-79 for more information about sound description structures).

If the media does not require a sample description, set this parameter to `nil`.

`numberOfSamples`

Specifies the number of samples contained in the reference. For details, see the `AddMediaSample` function description beginning on page 2-273.

`sampleFlags`

Contains flags that control the operation. The following flag is available (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 this flag to 0 if the sample is a sync sample.

`sampleTime`

Contains 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 do not want to receive this information, set this parameter to `nil`.

DESCRIPTION

The `AddMediaSampleReference` 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. As with the `AddMediaSample` function (described in the previous section), 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 the `BeginMediaEdits` function (described on page 2-271) before calling `AddMediaSampleReference`.

ERROR CODES

<code>invalidMedia</code>	-2008	This media is corrupted or invalid
Memory Manager errors		

SEE ALSO

If you want to add new samples to a media data file, use the `AddMediaSample` function, which is described in the previous section.

GetMediaSample

The `GetMediaSample` function returns a sample from a movie data file. You add samples to movie data files with the `AddMediaSample` function (described on page 2-273).

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

<code>theMedia</code>	Specifies the media for this operation. Your application obtains this media identifier from such Movie Toolbox functions as <code>NewTrackMedia</code> and <code>GetTrackMedia</code> (described on page 2-153 and page 2-206, respectively).
<code>dataOut</code>	Contains a handle. The <code>GetMediaSample</code> 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 <code>maxSizeToGrow</code> parameter.
<code>maxSizeToGrow</code>	Specifies the maximum number of bytes of sample data to be returned. The <code>GetMediaSample</code> function does not increase the handle specified by the <code>dataOut</code> 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.
<code>size</code>	Contains a pointer to a long integer. The <code>GetMediaSample</code> function updates the field referred to by the <code>size</code> parameter with the number of bytes of sample data returned in the handle specified by the <code>dataOut</code> parameter. Set this parameter to <code>nil</code> if you are not interested in this information.
<code>time</code>	Specifies the starting time of the sample to be retrieved. You must specify this value in the media's time scale.

Movie Toolbox

`sampleTime`

Contains a pointer to a time value. The `GetMediaSample` function updates this time value to indicate the actual time of the returned sample data. If you are not interested in this information, set this parameter to `nil`.

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.

`durationPerSample`

Contains 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 do not want this information.

`sampleDescriptionH`

Contains a handle to a sample description. The `GetMediaSample` function returns the sample description corresponding to the returned sample data. The function resizes this handle as appropriate. If you do not want the sample description, set this parameter to `nil`.

`sampleDescriptionIndex`

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

You can use this index to retrieve the sample description by calling the `GetMediaSampleDescription` function, which is described on page 2-226.

You can retrieve the sample description itself by using the `sampleDescriptionH` parameter.

`maxNumberOfSamples`

Specifies 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`

Contains 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 do not want this information, set this parameter to `nil`.

Movie Toolbox

sampleFlags

Contains a pointer to a short integer. The `GetMediaSample` function returns flags that describe the sample. The following flag is available (set unused flags to 0):

mediaSampleNotSync

Indicates that the sample that is returned is not a sync sample. Set this flag to 1 if the sample is not a sync sample. Set this flag to 0 if the sample is a sync sample.

If you do not want this information, set this parameter to `nil`.

ERROR CODES

`invalidMedia` -2008 This media is corrupted or invalid
File Manager errors
Memory Manager errors

GetMediaSampleReference

The `GetMediaSampleReference` function allows your application to obtain reference information about samples that are stored in a movie data file.

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

`theMedia` Specifies the media for this operation. Your application obtains this media identifier from such Movie Toolbox functions as `NewTrackMedia` and `GetTrackMedia` (described on page 2-153 and page 2-206, respectively).

`dataOffset` Contains a pointer to a long integer. The `GetMediaSampleReference` function 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.

Movie Toolbox

<code>size</code>	Contains a pointer to a long integer. The <code>GetMediaSampleReference</code> function updates the field referred to by the <code>size</code> parameter with the number of bytes of sample data referred to by the reference. Set this parameter to <code>nil</code> if you are not interested in this information.
<code>time</code>	Specifies the starting time of the sample reference to be retrieved. You must specify this value in the media's time scale.
<code>sampleTime</code>	<p>Contains a pointer to a time value. The <code>GetMediaSampleReference</code> function updates this time value to indicate the actual time of the returned sample data. If you are not interested in this information, set this parameter to <code>nil</code>.</p> <p>The returned time may differ from the time you specified with the <code>time</code> parameter. This will occur if the time you specified falls in the middle of a sample.</p>
<code>durationPerSample</code>	Contains 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 do not want this information.
<code>sampleDescriptionH</code>	Contains a handle to a sample description. The <code>GetMediaSampleReference</code> function returns the sample description corresponding to the returned sample data. The function resizes this handle as appropriate. If you do not want the sample description, set this parameter to <code>nil</code> .
<code>sampleDescriptionIndex</code>	<p>Contains a pointer to a long integer. The <code>GetMediaSampleReference</code> function returns an index value to the sample description that corresponds to the returned sample data. You can use this index to retrieve the media sample description with the <code>GetMediaSampleDescription</code> function, which is described on page 2-226. If you do not want this information, set this parameter to <code>nil</code>.</p> <p>You can retrieve the sample description itself by using the <code>sampleDescriptionH</code> parameter.</p>
<code>maxNumberOfSamples</code>	<p>Specifies 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.</p> <p>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 <code>numberOfSamples</code> parameter.</p>
<code>numberOfSamples</code>	Contains a pointer to a long integer. The <code>GetMediaSampleReference</code> function updates the field referred to by this parameter with the number of samples referred to by the returned reference. If you do not want this information, set this parameter to <code>nil</code> .

Movie Toolbox

`sampleFlags`

Contains a pointer to a short integer. The `GetMediaSampleReference` function returns flags that describe the samples referred to by the reference. The following flag is available (unused flags are set to 0):

`mediaSampleNotSync`

Indicates the sample that is returned is not a sync sample. Set this flag to 1 if the sample is not a sync sample. Set this flag to 0 if the sample is a sync sample.

If you do not want this information, set this parameter to `nil`.

DESCRIPTION

The `GetMediaSampleReference` function is similar to `GetMediaSample`, except that it does not return the sample data.

ERROR CODES

<code>invalidMedia</code>	-2008	This media is corrupted or invalid
Memory Manager errors		

Media Functions

The Movie Toolbox does not contain any support for specific media types. Rather, it delegates this work to media handler components. The Movie Toolbox provides a number of functions that allow your application to interact with media handlers. This section describes those functions. It has been divided into the following topics:

- “Selecting Media Handlers” describes the functions that you can use to gain access to a media handler
- “Video Media Handler Functions” describes the functions that allow your application to interact with video media handlers
- “Sound Media Handler Functions” describes the functions that allow your application to interact with sound media handlers
- “Text Media Handler Functions” describes the functions that allow your application to interact with text media handlers

Selecting Media Handlers

Media handler components are responsible for interpreting and manipulating a media's sample data. Each type of media has its own media handler, which deals with the specific characteristics of the media data. The Movie Toolbox provides a set of functions that allow you to gather information about a media handler and assign a particular media handler to a media. This section discusses those functions.

Each media handler has an associated data handler for each data reference. The data handler is responsible for fetching, storing, and caching the data that the media handler uses. The Movie Toolbox provides functions that allow you to get information about data handlers and to assign a particular data handler to a media.

The `GetMediaHandler` and `GetMediaHandlerDescription` functions allow you to retrieve information about a media handler.

You can use the `SetMediaHandler` function to assign a media handler to a media.

The `GetMediaDataHandler` and `GetMediaDataHandlerDescription` functions enable you to retrieve information about a data handler. Use the `SetMediaDataHandler` function to assign a data handler to a media.

GetMediaHandlerDescription

The `GetMediaHandlerDescription` function allows your application to retrieve information about a media handler. You specify the media.

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

theMedia Specifies the media for this operation. Your application obtains this media identifier from such Movie Toolbox functions as `NewTrackMedia` and `GetTrackMedia` (described on page 2-153 and page 2-206, respectively).

mediaType Contains a pointer to a field of data type `OSType`. The Movie Toolbox returns the media type identifier. 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 do not want to receive this information, set the `mediaType` parameter to `nil`. The following values are available:

<code>VideoMediaType</code>	Video media
<code>SoundMediaType</code>	Sound media
<code>TextMediaType</code>	Text media

Movie Toolbox

`creatorName`

Points to a string. The Movie Toolbox returns the name of the media handler's creator. If you do not want to receive this information, set this parameter to `nil`.

`creatorManufacturer`

Contains a pointer to a long integer. The Movie Toolbox returns the 4-byte value that identifies the manufacturer of the component. If you do not want to retrieve this information, set this parameter to `nil`.

DESCRIPTION

The Movie Toolbox returns information about that media's media handler. This information describes the media handler that created the media, not the handler that is currently assigned to the media.

ERROR CODES

`invalidMedia` `-2008` This media is corrupted or invalid

GetMediaHandler

The `GetMediaHandler` function allows you to obtain a reference to a media handler component.

You can use this reference to call the media handler directly. See “Video Media Handler Functions,” which begins on page 2-287, and “Sound Media Handler Functions,” which begins on page 2-288, for information about the functions that are supported by video and sound media handlers.

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

`theMedia` Specifies the media for this operation. Your application obtains this media identifier from such Movie Toolbox functions as `NewTrackMedia` and `GetTrackMedia` (described on page 2-153 and page 2-206, respectively).

DESCRIPTION

The `GetMediaHandler` function returns a reference to the media's media handler. If the function could not locate the media handler, it sets this reference to `nil`. You can use this reference to call the media handler.

ERROR CODES

`invalidMedia` `-2008` This media is corrupted or invalid

SetMediaHandler

The `SetMediaHandler` function allows you to assign a specific media handler to a track. 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.

```
pascal OSErr SetMediaHandler (Media theMedia,
                             MediaHandlerComponent mH);
```

<code>theMedia</code>	Specifies the track for this operation. Your application obtains this track identifier from such Movie Toolbox functions as <code>NewMovieTrack</code> and <code>GetMovieTrack</code> (described on page 2-151 and page 2-204, respectively).
<code>mH</code>	Contains a reference to a media handler component. You obtain this reference from the <code>GetMediaHandler</code> function, which is described in the previous section.

Note

Your application should not need to call the `SetMediaHandler` function. The Movie Toolbox assigns a media handler to each track when you load a movie. ♦

ERROR CODES

<code>invalidHandler</code>	-2013	This handler is invalid
-----------------------------	-------	-------------------------

GetMediaDataHandlerDescription

The `GetMediaDataHandlerDescription` function allows your application to retrieve information about a media's data handler. You specify the media.

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

<code>theMedia</code>	Specifies the media for this operation. Your application obtains this media identifier from such Movie Toolbox functions as <code>NewTrackMedia</code> and <code>GetTrackMedia</code> (described on page 2-153 and page 2-206, respectively).
<code>index</code>	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.

Movie Toolbox

dhType	Contains 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 do not want to receive this information, set the dhType parameter to nil.
creatorName	Points to a string. The Movie Toolbox returns the name of the data handler's creator. If you do not want to receive this information, set this parameter to nil.
creatorManufacturer	Contains a pointer to a long integer. The Movie Toolbox returns the 4-byte value that identifies the manufacturer of the component. If you do not want to retrieve this information, set this parameter to nil.

DESCRIPTION

The Movie Toolbox returns information about that media's data handler. This information describes the data handler that created the media data, not the handler that is currently assigned to the media.

ERROR CODES

invalidMedia	-2008	This media is corrupted or invalid
--------------	-------	------------------------------------

GetMediaDataHandler

The GetMediaDataHandler function allows you to determine a media's data handler.

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

theMedia	Specifies the media for this operation. Your application obtains this media identifier from such Movie Toolbox functions as NewTrackMedia and GetTrackMedia (described on page 2-153 and page 2-206, respectively).
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.

Movie Toolbox

DESCRIPTION

The `GetMediaDataHandler` function returns a data handler identifier. This identifier is a component instance that specifies a connection to a data handler component (see the chapter “Component Manager” in *Inside Macintosh: More Macintosh Toolbox* for more information about components). If the Movie Toolbox cannot determine the data handler for the media you specify, the function sets this returned value to `nil`.

Note

Your application should not need to call this function. ♦

ERROR CODES

<code>invalidMedia</code>	-2008	This media is corrupted or invalid
---------------------------	-------	------------------------------------

SetMediaDataHandler

The `SetMediaDataHandler` function allows you to assign a data handler to a media.

```
pascal OSErr SetMediaDataHandler (Media theMedia, short index,
                                   DataHandlerComponent dataHandler);
```

<code>theMedia</code>	Specifies the media for this operation. Your application obtains this media identifier from such Movie Toolbox functions as <code>NewTrackMedia</code> and <code>GetTrackMedia</code> (described on page 2-153 and page 2-206, respectively).
-----------------------	---

<code>index</code>	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.
--------------------	--

<code>dataHandler</code>	Specifies the data handler for the media. This identifier is a component instance that specifies a connection to a data handler component (see the chapter “Component Manager” in <i>Inside Macintosh: More Macintosh Toolbox</i> for more information about components). 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.
--------------------------	---

DESCRIPTION

When you create a new media or load an existing media into memory, the media handler assigns an appropriate data handler to the track’s media.

Note

Your application should not call the `SetMediaDataHandler` function. The Movie Toolbox assigns a data handler to each media when you load a movie. ♦

ERROR CODES

badComponentType	-2005	Component cannot accommodate this data
invalidMedia	-2008	This media is corrupted or invalid

Video Media Handler Functions

Video media handlers are responsible for interpreting and manipulating video data. These media handlers allow you to call them directly to work with some graphics settings. This section describes the functions supported by video media handlers.

Video media handlers maintain a graphics mode and color value that affect the display of video data. You can use the `SetVideoMediaGraphicsMode` and `GetVideoMediaGraphicsMode` functions to work with these characteristics. See *Inside Macintosh: Imaging* for more information about setting color values for use with the `addPin`, `subPin`, `blend`, and `transparent` drawing modes.

Sample descriptions for video media are stored in image description structures. For a complete discussion of the format and content of the image description structure, see the chapter “Image Compression Manager” in this book.

SetVideoMediaGraphicsMode

The `SetVideoMediaGraphicsMode` function allows you to set the graphics mode and blend color of a video media.

```
pascal HandlerError SetVideoMediaGraphicsMode (MediaHandler mh,
                                                long graphicsMode,
                                                const RGBColor *opColor);
```

mh	Contains a reference to a media handler. You obtain this reference from the <code>GetMediaHandler</code> function, which is described on page 2-283.
graphicsMode	Specifies the graphics mode of the media handler. This is a <code>QuickDraw</code> transfer mode value.
opColor	Contains a pointer to the color for use in blending and transparent operations. The media handler passes this color to <code>QuickDraw</code> as appropriate when you draw in <code>addPin</code> , <code>subPin</code> , <code>blend</code> , or <code>transparent</code> mode.

ERROR CODES

Component Manager errors

SEE ALSO

You can retrieve the graphics mode and blend color currently in use by a video media handler by calling the `GetVideoMediaGraphicsMode` function, which is described in the next section.

GetVideoMediaGraphicsMode

The `GetVideoMediaGraphicsMode` function allows you to obtain the graphics mode and blend color values currently in use by a video media handler.

```
pascal HandlerError GetVideoMediaGraphicsMode (MediaHandler mh,
                                                long *graphicsMode,
                                                RGBColor *opColor);
```

<code>mh</code>	Contains a reference to a media handler. You obtain this reference from the <code>GetMediaHandler</code> function, which is described on page 2-283.
<code>graphicsMode</code>	Contains a pointer to a long integer. The media handler returns the graphics mode currently in use by the media handler. This is a QuickDraw transfer mode value.
<code>opColor</code>	Contains a pointer to an RGB color structure. The Movie Toolbox returns the color currently in use by the media handler. This is the blend value for blends and the transparent color for transparent operations. The Movie Toolbox supplies this value to QuickDraw when you draw in <code>addPin</code> , <code>subPin</code> , <code>blend</code> , or <code>transparent</code> mode.

ERROR CODES

Component Manager errors

SEE ALSO

You can set the graphics mode and blend color of a video media handler by calling the `SetVideoMediaGraphicsMode` function, which is described in the previous section.

Sound Media Handler Functions

Sound media handlers are responsible for interpreting and manipulating sound data. These media handlers allow you to call them directly to work with some audio settings. This section describes the functions supported by sound media handlers.

Sound media handlers maintain balance information for their audio data. You can use the `SetSoundMediaBalance` and `GetSoundMediaBalance` functions to work with a handler's balance setting.

Sample descriptions for sound media are stored in sound description structures. See “The Sound Description Structure” on page 2-79 for a discussion of the format and content of the sound description structure.

SetSoundMediaBalance

The `SetSoundMediaBalance` function sets the balance of a sound media.

```
pascal HandlerError SetSoundMediaBalance (MediaHandler mh,
                                           short balance);
```

mH	Contains a reference to a media handler. You obtain this reference from the <code>GetMediaHandler</code> function, which is described on page 2-283.
balance	Specifies the balance setting of the media handler as a 16-bit, fixed-point value. The high-order 8 bits contain the integer part of the value; the low-order 8 bits contain the fractional part. Valid balance values range from -1.0 to 1.0. Negative values emphasize the left sound channel, and positive values emphasize the right sound channel; a value of 0 specifies neutral balance.

ERROR CODES

Component Manager errors

GetSoundMediaBalance

The `GetSoundMediaBalance` function returns the balance of a sound media.

```
pascal HandlerError GetSoundMediaBalance (MediaHandler mh,
                                           short *balance);
```

mH	Contains a reference to a media handler. You obtain this reference from the <code>GetMediaHandler</code> function, which is described on page 2-283.
balance	Contains a pointer to an integer. The Movie Toolbox returns the current balance setting of the media handler as a 16-bit, fixed-point value. The high-order 8 bits contain the integer part of the value; the low-order 8 bits contain the fractional part. Valid balance values range from -1.0 to 1.0. Negative values emphasize the left sound channel, and positive values emphasize the right sound channel; a value of 0 specifies neutral balance.

ERROR CODES

Component Manager errors