Data Components Reference for QuickTime

QuickTime > Import & Export

2006-05-23

Contents

Data Components Reference for QuickTime

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

Overview

Data components allow applications to place various types of data into a QuickTime movie or extract data from a movie in a specified format.

Functions by Task

Data Codec Functions

`DataCodecBeginInterruptSafe` (page 13)
: Called before performing a compression or decompression operation during interrupt time.

`DataCodecCompress` (page 14)
: Compresses data using the specified compressor component.

`DataCodecDecompress` (page 16)
: Decompresses data using the specified compressor component.

`DataCodecEndInterruptSafe` (page 18)
: Releases resources used by DataCodecBeginInterruptSafe.

Identifying Data References

`DataHCompareDataRef` (page 24)
: Compares a supplied data reference against its current data reference and returns a Boolean value indicating whether the data references are equivalent (that is, the two data references identify the same container).

`DataHGetDataRef` (page 34)
: Retrieves your component's current data reference.

`DataHResolveDataRef` (page 59)
: Locates the container associated with a given data reference.

`DataHSetDataRef` (page 63)
: Assigns a data reference to your data handler component.

Managing Data Handler Components

`DataHFlushCache` (page 29)
: Discards the contents of any cached read buffers.

`DataHFlushData` (page 29)
: Forces any data in your component's write buffers to be written to the device that contains the current data reference.

`DataHPlaybackHints` (page 52)
: Provides additional information to your component that you may use to optimize the operation of your data handler.

`DataHTask` (page 70)
: Cedes processor time to your data handler.

Reading Movie Data

`DataHCloseForRead` (page 22)
: Closes read-only access to its data reference.

`DataHFinishData` (page 28)
: Completes or cancels one or more queued read requests.

`DataHGetAvailableFileSize` (page 30)
: Returns the available file size for a data handler component.

`DataHGetData` (page 31)
: Reads data from its current data reference, which is a synchronous read operation.

`DataHGetFileSize` (page 38)
: Returns the size, in bytes, of the current data reference.

`DataHGetScheduleAheadTime` (page 48)
: Reports how far in advance it prefers clients to issue read requests.

`DataHOpenForRead` (page 50)
: Opens a data handler's current data reference for read-only access.

`DataHScheduleData` (page 60)
: Reads data from its current data reference, which can be a synchronous read operation or an asynchronous read operation.

Selecting a Data Handler

`DataHCanUseDataRef` (page 21)
: Reports whether a data handler can access the data associated with a specified data reference.

`DataHGetDeviceIndex` (page 36)
: Returns a value that identifies the device on which a data reference resides.

`DataHGetVolumeList` (page 49)
: Returns a list of the volumes your component can access, along with flags indicating your component's capabilities for each volume.

Using Data References to Access Media

`DataHDeleteFile` (page 27)
    Deletes a data handler's data storage file.

`DataHGetInfo` (page 42)
    Retrieves information from a data handler.

`DataHSetMovieUsageFlags` (page 68)
    Sets the way that a data handler appends data to its storage.

Working With The Idle Manager

`DataHSetIdleManager` (page 67)
    Lets a data handler report its idling needs.

Writing Movie Data

`DataHCloseForWrite` (page 23)
    Closes write-only access to its data reference.

`DataHCreateFile` (page 25)
    Creates a new container that meets the specifications of the current data reference.

`DataHGetFreeSpace` (page 41)
    Reports the number of bytes available on the device that contains the current data reference.

`DataHGetPreferredBlockSize` (page 47)
    Reports the block size that it prefers to use when accessing the current data reference.

`DataHOpenForWrite` (page 51)
    Opens your component's current data reference for write-only access.

`DataHPreextend` (page 55)
    Allocates new space for the current data reference, enlarging the container.

`DataHPutData` (page 56)
    Writes data to a component's current data reference.

`DataHSetFileSize` (page 65)
    Sets the size, in bytes, of the current data reference.

`DataHWrite` (page 71)
    Writes data to its current data reference.

Supporting Functions

`DataCodecCompressPartial` (page 15)
    Undocumented

`DataCodecDecompressPartial` (page 17)
    Undocumented

`DataCodecGetCompressBufferSize` (page 19)
: Returns the maximum possible size of the compressed data that will be returned using the specified compressor component.

`DataHAddMovie` (page 20)
: Assigns movie data to a data handler.

`DataHAppend64` (page 20)
: Appends data to the current data reference.

`DataHCreateFileWithFlags` (page 26)
: Undocumented

`DataHDoesBuffer` (page 27)
: Reports whether a data handler does buffer reads and writes.

`DataHGetCacheSizeLimit` (page 30)
: Returns the cache size limit for a data handler component.

`DataHGetDataAvailability` (page 32)
: Undocumented

`DataHGetDataInBuffer` (page 33)
: Returns the information about the data in a data handler component's buffer.

`DataHGetDataRate` (page 33)
: Undocumented

`DataHGetDataRefAsType` (page 34)
: Retrieves a data handler component's current data reference of a given type.

`DataHGetDataRefExtension` (page 35)
: Retrieves your component's current data reference extension data.

`DataHGetDataRefWithAnchor` (page 36)
: Retrieves a data handler component's component's current data reference and anchor data reference.

`DataHGetFileName` (page 37)
: Retrieves the name of the file supplying the current data reference for a data handler.

`DataHGetFileSize64` (page 39)
: Provides a 64-bit version of DataHGetFileSize.

`DataHGetFileSizeAsync` (page 39)
: Returns the size of the current data reference, invoking a completion callback.

`DataHGetFileTypeOrdering` (page 40)
: Returns the preferred ordering for file typing information.

`DataHGetFreeSpace64` (page 41)
: Provides a 64-bit version of DataHGetFreeSpace.

`DataHGetInfoFlags` (page 43)
: Provides information about the operation of a data handler component.

`DataHGetMacOSFileType` (page 43)
: Gets the Mac OS file type for a data handler's current data reference.

`DataHGetMIMEType` (page 44)
: Gets the MIME type for a data handler's current data reference.

`DataHGetMIMETypeAsync` (page 44)
: Performs asynchronous discovery of a HTTP/FTP connection's MIME type.

`DataHGetMovie` (page 45)
: Gets the movie for a data handler's current data reference.

`DataHGetMovieWithFlags` (page 46)
: Gets the movie for a data handler's current data reference, allowing the flags that would be passed to NewMovieFromDataRef to be passed to the handler.

`DataHGetTemporaryDataRefCapabilities` (page 48)
: Undocumented

`DataHIsStreamingDataHandler` (page 50)
: Determines if a data handler handles streaming data.

`DataHPlaybackHints64` (page 53)
: Provides a 64-bit version of DataHPlaybackHints.

`DataHPollRead` (page 54)
: Undocumented

`DataHPreextend64` (page 55)
: Provides a 64-bit version of DataHPreextend.

`DataHReadAsync` (page 57)
: Undocumented

`DataHRenameFile` (page 58)
: Undocumented

`DataHScheduleData64` (page 61)
: Provides a 64-bit version of DataHScheduleData.

`DataHSetCacheSizeLimit` (page 62)
: Sets the cache size limit for a data handler component.

`DataHSetDataRefExtension` (page 64)
: Sets your component's current data reference extension data.

`DataHSetDataRefWithAnchor` (page 65)
: Sets the data reference and anchor data reference for a data handler.

`DataHSetFileSize64` (page 66)
: Provides a 64-bit version of DataHSetFileSize.

`DataHSetMacOSFileType` (page 67)
: Sets the Mac OS file type for a data handler's current data reference.

`DataHSetTimeBase` (page 68)
: Sets the time base for a data handler component.

`DataHSetTimeHints` (page 69)
: Undocumented

`DataHUpdateMovie` (page 70)
: Updates the movie for a data handler's current data reference.

`DataHUseTemporaryDataRef` (page 71)
: Undocumented

`DataHWrite64` (page 73)
: Provides a 64-bit version of DataHWrite.

`DisposeCDataHandlerUPP` (page 74)
: Disposes of a CDataHandlerUPP pointer.

`DisposeCharDataHandlerUPP` (page 74)
    Disposes of a CharDataHandlerUPP pointer.

`DisposeCommentHandlerUPP` (page 75)
    Disposes of a CommentHandlerUPP pointer.

`DisposeDataHCompletionUPP` (page 75)
    Disposes of a DataHCompletionUPP pointer.

`DisposeEndDocumentHandlerUPP` (page 76)
    Disposes of an EndDocumentHandlerUPP pointer.

`DisposeEndElementHandlerUPP` (page 76)
    Disposes of an EndElementHandlerUPP pointer.

`DisposePreprocessInstructionHandlerUPP` (page 76)
    Disposes of a PreprocessInstructionHandlerUPP pointer.

`DisposeStartDocumentHandlerUPP` (page 77)
    Disposes of a StartDocumentHandlerUPP pointer.

`DisposeStartElementHandlerUPP` (page 77)
    Disposes of a StartElementHandlerUPP pointer.

`DisposeVdigIntUPP` (page 77)
    Disposes of a VdigIntUPP pointer.

`NewCDataHandlerUPP` (page 78)
    Allocates a Universal Procedure Pointer for the CDataHandlerProc callback.

`NewCharDataHandlerUPP` (page 78)
    Undocumented

`NewCommentHandlerUPP` (page 79)
    Undocumented

`NewDataHCompletionUPP` (page 79)
    Allocates a Universal Procedure Pointer for the DataHCompletionProc callback.

`NewEndDocumentHandlerUPP` (page 80)
    Undocumented

`NewEndElementHandlerUPP` (page 80)
    Undocumented

`NewPreprocessInstructionHandlerUPP` (page 81)
    Undocumented

`NewStartDocumentHandlerUPP` (page 81)
    Undocumented

`NewStartElementHandlerUPP` (page 82)
    Undocumented

`NewVdigIntUPP` (page 82)
    Allocates a Universal Procedure Pointer for the VdigIntProc callback.

`XMLParseAddAttribute` (page 83)
    Undocumented

`XMLParseAddAttributeAndValue` (page 83)
    Undocumented

`XMLParseAddAttributeValueKind` (page 84)
    Undocumented

Functions

DataCodecBeginInterruptSafe

Called before performing a compression or decompression operation during interrupt time.

```
ComponentResult DataCodecBeginInterruptSafe (
   DataCodecComponent dc,
   unsigned long maxSrcSize
);
```

Parameters

dc

The instance of a compressor or decompressor component for this request. Your software obtains this reference when calling the Component Manager's `OpenComponent` or `OpenDefaultComponent` function.

maxSrcSize

The maximum size of a block of data to be compressed or decompressed, in bytes.

Return Value

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

Discussion

This function allocates any temporary buffers that are necessary to perform the operation during interrupt time. To release the resources used to make the operation safe during interrupt time, call the `DataCodecEndInterruptSafe` (page 18) function or close the instance of the compressor or decompressor component.

Special Considerations

If the function returns an error, your software must not perform compression or decompression operations during interrupt time.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`QuickTimeComponents.h`

DataCodecCompress

Compresses data using the specified compressor component.

```
ComponentResult DataCodecCompress (
   DataCodecComponent dc,
   void *srcData,
   UInt32 srcSize,
   void *dstData,
   UInt32 dstBufferSize,
   UInt32 *actualDstSize,
   UInt32 *decompressSlop
);
```

Parameters

dc

The compressor component to used.

srcData

A pointer to the data to be compressed.

srcSize
: The size of the data to be compressed, in bytes.

dstData
: A pointer to the buffer in which to store the compressed data.

dstBufferSize
: The size of the buffer in which to store the compressed data, in bytes.

actualDstSize
: The size of the compressed data that was created, in bytes.

decompressSlop
: The number of bytes that should be added to the decompression buffer size if decompression occurs in place.

Return Value

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

Discussion

Before calling this function, you should call `DataCodecGetCompressBufferSize` (page 19) to obtain the maximum possible size of the compressed data that will be returned. You can then use this value as the value of the `dstBufferSize` parameter. Note that a buffer for compressed data that is the same size as the uncompressed data may not be large enough: in some cases, the size of the compressed data can be larger than the size of the decompressed data. When you compress data, you should store the size of data before compression at the beginning of the file, immediately before the compressed data. This allows you to obtain the size of the decompressed data and allocate the buffer for storing the decompressed data before calling `DataCodecDecompress` (page 16).

Special Considerations

You can compress sprites by calling this function. If you do, you must include the uncompressed size of the sample at the beginning of the sample, before the compressed data, and store the `component` subtype of the data compressor used to compress the sprite in the `decompressorType` field of the sample's `SpriteDescription` structure. You can also compress QuickDraw 3D media samples by calling this function.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

qtactiontargets

qtactiontargets.win

qtsprites.win

qtwiredsprites.win

WiredSprites

Declared In

`QuickTimeComponents.h`

DataCodecCompressPartial

Undocumented

```
ComponentResult DataCodecCompressPartial (
   DataCodecComponent dc,
   void **next_in,
   unsigned long *avail_in,
   unsigned long *total_in,
   void **next_out,
   unsigned long *avail_out,
   unsigned long *total_out,
   Boolean tryToFinish,
   Boolean *didFinish
);
```

Parameters

`dc`

The compressor component to used.

`next_in`

Undocumented

`avail_in`

Undocumented

`total_in`

Undocumented

`next_out`

Undocumented

`avail_out`

Undocumented

`total_out`

Undocumented

`tryToFinish`

Undocumented

`didFinish`

Undocumented

Return Value

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

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`QuickTimeComponents.h`

DataCodecDecompress

Decompresses data using the specified compressor component.

```
ComponentResult DataCodecDecompress (
   DataCodecComponent dc,
   void *srcData,
   UInt32 srcSize,
   void *dstData,
   UInt32 dstBufferSize
);
```

Parameters

dc

The decompressor component to used.

srcData

A pointer to the data to be decompressed.

srcSize

The size of the data to be decompressed, in bytes.

dstData

A pointer to the buffer in which to store the decompressed data.

dstBufferSize

The size of the buffer in which to store the decompressed data, in bytes.

Return Value

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

Discussion

Before allocating the buffer in which to store decompressed data, you need to get the size of the decompressed data. The size is normally stored at the beginning of the file, immediately before the compressed data.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`QuickTimeComponents.h`

DataCodecDecompressPartial

Undocumented

```
ComponentResult DataCodecDecompressPartial (
   DataCodecComponent dc,
   void **next_in,
   unsigned long *avail_in,
   unsigned long *total_in,
   void **next_out,
   unsigned long *avail_out,
   unsigned long *total_out,
   Boolean *didFinish
);
```

Parameters

dc

The decompressor component to used.

next_in

Undocumented

avail_in

Undocumented

total_in

Undocumented

next_out

Undocumented

avail_out

Undocumented

total_out

Undocumented

didFinish

Undocumented

Return Value

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

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`QuickTimeComponents.h`

DataCodecEndInterruptSafe

Releases resources used by DataCodecBeginInterruptSafe.

```
ComponentResult DataCodecEndInterruptSafe (
   DataCodecComponent dc
);
```

Parameters

dc

The instance of a compressor or decompressor component for this request.

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

Discussion
When your software has finished a compression or decompression operation that must be performed during interrupt time, it can call this function to release any memory of other resources that were used by `DataCodecBeginInterruptSafe`.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`QuickTimeComponents.h`

DataCodecGetCompressBufferSize

Returns the maximum possible size of the compressed data that will be returned using the specified compressor component.

```
ComponentResult DataCodecGetCompressBufferSize (
   DataCodecComponent dc,
   UInt32 srcSize,
   UInt32 *dstSize
);
```

Parameters

dc

The compressor component to used.

srcSize

The size in bytes of the data being compressed.

dstSize

A pointer to the maximum size of the compressed data that will be returned.

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

Discussion
The actual size of the compressed data will likely be smaller than the size you initially have to allocate, so after you compress the data you should shrink the compressed data handle down to the actual data size. When compressing a movie, allocate an extra ten 32-bit integers of space to store the compressed movie resource header information, as shown in the following code sample:

```
unsigned long compressedSize;
Handle compressedData;
DataCodecGetCompressBufferSize(dataCompressor, movieResourceSize,
&compressedSize);
compressedData =NewHandle(compressedSize + (10 * sizeof(UInt32)));
```

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
qtactiontargets

qtactiontargets.win

qtsprites.win

qtwiredsprites.win

WiredSprites

Declared In
`QuickTimeComponents.h`

DataHAddMovie

Assigns movie data to a data handler.

```
ComponentResult DataHAddMovie (
   DataHandler dh,
   Movie theMovie,
   short *id
);
```

Parameters

dh

A data handler component.

theMovie

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

id

A pointer to the field that specifies the resource containing the movie data that is to be added.

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

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`QuickTimeComponents.h`

DataHAppend64

Appends data to the current data reference.

```
ComponentResult DataHAppend64 (
   DataHandler dh,
   void *data,
   wide *fileOffset,
   unsigned long size
);
```

Parameters

dh

A data handler component.

data

A pointer to data to be appended.

fileOffset

A pointer to a 64-bit value that represents the offset in the file of data to be appended.

size

The size of the data to be appended.

Return Value

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

Version Notes

Introduced in QuickTime 4.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`QuickTimeComponents.h`

DataHCanUseDataRef

Reports whether a data handler can access the data associated with a specified data reference.

```
ComponentResult DataHCanUseDataRef (
   DataHandler dh,
   Handle dataRef,
   long *useFlags
);
```

Parameters

dh

Identifies the calling program's connection to your data handler component.

dataRef

The data reference. This parameter contains a handle to the information that identifies the container in question.

useFlags

A pointer to a field of flags (see below) that your data handler component uses to indicate its ability to access the container identified by the `dataRef` parameter. Set all appropriate flags to 1; set unused flags to 0. For example, if your component supports networked multimedia servers using a special set of protocols, your data handler should set the `kDataHCanRead` and `kDataHCanSpecialRead` flags to 1 for any container that is on that server. In addition, if your component can write to the server, set the `kDataHCanWrite` and `kDataHCanSpecialWrite` flags to 1 (perhaps along with `kDataHCanStreamingWrite`). If your data handler cannot access the container, set the whole field to 0. See these constants:

`kDataHCanRead`

`kDataHSpecialRead`

`kDataHSpecialReadFile`

`kDataHCanWrite`

`kDataHSpecialWrite`

`kDataHCanStreamingWrite`

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

Discussion
Apple's standard data handler sets both the `kDataHCanRead` and `kDataHCanWrite` flags to 1 for any data reference it receives, indicating that it can read from and write to any volume.

Special Considerations

Your data handler may use any facilities necessary to determine whether it can access the container. Bear in mind, though, that your component should try to be as quick about this determination as possible, in order to minimize the chance that the delay will be noticed by the user.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`QuickTimeComponents.h`

DataHCloseForRead

Closes read-only access to its data reference.

```
ComponentResult DataHCloseForRead (
   DataHandler dh
);
```

Parameters

dh

Identifies the calling program's connection to your data handler component.

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

Discussion

Data handler components provide two basic read facilities: `DataHGetData` (page 31) function is a fully synchronous read operation, while `DataHScheduleData` (page 60) is asynchronous. Applications provide scheduling information when they call your component's `DataHScheduleData` function. When your component processes the queued request, it calls the application's data-handler completion function. Before any application can read data from a data reference, it must open read access to that reference by calling your component's `DataHOpenForRead` (page 50) function. `DataHCloseForRead` (page 22) closes that read access path.

Special Considerations

Note that a client program may close its connection to your component (by calling `CloseComponent`) without closing the read path. If this happens, your component should close the data reference before closing the connection.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

qtdataref

qtdataref.win

qtfiletransfer

ThreadsImporter

ThreadsImportMovie

Declared In

`QuickTimeComponents.h`

DataHCloseForWrite

Closes write-only access to its data reference.

```
ComponentResult DataHCloseForWrite (
   DataHandler dh
);
```

Parameters

dh

Identifies the calling program's connection to your data handler component.

Return Value

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

Discussion

Data handlers provide two distinct write facilities: `DataHPutData` (page 56) is a simple synchronous interface that allows applications to append data to the end of a container, and `DataHWrite` is a more capable, asynchronous write function that is suitable for movie capture operations. Before writing data to a data reference, applications must call your component's `DataHOpenForWrite` (page 51) function to open a write path to the container. `DataHCloseForWrite` closes that write path. As is the case with `DataHScheduleData` (page 60), your component calls the application's data-handler completion function when you are done with the write request.

Special Considerations

A client program may close its connection to your component (by calling `CloseComponent`) without closing the write path. If this happens, your component should close the data reference before closing the connection. Note that some data handlers may not support write operations. For example, some shared devices, such as a CD-ROM "jukebox," may be read-only devices. As a result, it is very important that your data handler correctly report its write capabilities to client programs.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
qtdataref

qtdataref.win

qtfiletransfer

ThreadsImporter

ThreadsImportMovie

Declared In
`QuickTimeComponents.h`

DataHCompareDataRef

Compares a supplied data reference against its current data reference and returns a Boolean value indicating whether the data references are equivalent (that is, the two data references identify the same container).

```
ComponentResult DataHCompareDataRef (
   DataHandler dh,
   Handle dataRef,
   Boolean *equal
);
```

Parameters

dh

Identifies the calling program's connection to your data handler component.

dataRef

The data reference to be compared to your component's current data reference. Different types of containers may require different types of data references. For example, a reference to a memory-based movie may be a handle, while a reference to a file-based movie may be an alias. Apple's memory-based data handler for the Macintosh uses handles (and has a subtype value of `'hndl'`), while the HFS data handler uses Alias Manager aliases (its subtype value is `'alis'`). See `Data References`.

equal

A pointer to a Boolean. Your component should set that Boolean to TRUE if the two data references identify the same container. Otherwise, set the Boolean to FALSE.

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

Discussion
All data handler components use data references to identify and locate a movie's container. `DataHCompareDataRef` asks your component to compare a data reference against the current data reference and indicate whether the references are equivalent (that is, refer to the same container). Client programs can correlate data references with data handlers by matching the component's subtype value with the `data` reference type; the subtype value indicates the type of data reference the component supports. All data handlers with the same subtype value must support the same data reference type.

Special Considerations

Note that your component cannot simply compare the bits in the two data references. For example, two completely different aliases may refer to the same HFS file. Consequently, you need to completely resolve the data reference in order to determine the file identified by the reference.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`QuickTimeComponents.h`

DataHCreateFile

Creates a new container that meets the specifications of the current data reference.

```
ComponentResult DataHCreateFile (
   DataHandler dh,
   OSType creator,
   Boolean deleteExisting
);
```

Parameters

dh

Identifies the calling program's connection to your data handler component.

creator

The creator type of the new container. If the client program sets this parameter to 0, your component should choose a reasonable value (for example, 'TV0D', the `creator` type for Apple's movie player).

deleteExisting

Indicates whether to delete any existing data. If this parameter is set to TRUE and a container already exists for the current data reference, your component should delete that data before creating the new container. If this parameter is set to FALSE, your component should preserve any data that resides in the container defined by the current data reference (if there is any).

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

Discussion
Data handlers provide two distinct write facilities: `DataHPutData` (page 56) is a simple synchronous interface that allows applications to append data to the end of a container, while `DataHWrite` is a more capable, asynchronous write function that is suitable for movie capture operations. As is the case with `DataHScheduleData` (page 60), your component calls the application's data-handler completion function when you are done with the write request.

Special Considerations

Note that some data handlers may not support write operations. For example, some shared devices, such as a CD-ROM "jukebox," may be read-only devices. As a result, it is very important that your data handler correctly report its write capabilities to client programs.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
ElectricImageComponent

ElectricImageComponent.win

Declared In
`QuickTimeComponents.h`

DataHCreateFileWithFlags

Undocumented

```
ComponentResult DataHCreateFileWithFlags (
   DataHandler dh,
   OSType creator,
   Boolean deleteExisting,
   UInt32 flags
);
```

Parameters

`dh`

Identifies the calling program's connection to your data handler component.

`creator`

The creator type of the new container. If the client program sets this parameter to 0, your component should choose a reasonable value (for example, 'TV0D', the `creator` type for Apple's movie player).

`deleteExisting`

Indicates whether to delete any existing data. If this parameter is set to TRUE and a container already exists for the current data reference, your component should delete that data before creating the new container. If this parameter is set to FALSE, your component should preserve any data that resides in the container defined by the current data reference (if there is any).

`flags`

Undocumented

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

Version Notes
Introduced in QuickTime 5.

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

Declared In
`QuickTimeComponents.h`

DataHDeleteFile

Deletes a data handler's data storage file.

```
ComponentResult DataHDeleteFile (
   DataHandler dh
);
```

Parameters

dh

A data handler component.

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

Version Notes
Introduced in QuickTime 6.

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

Related Sample Code
QTExtractAndConvertToMovieFile

Declared In
`QuickTimeComponents.h`

DataHDoesBuffer

Reports whether a data handler does buffer reads and writes.

```
ComponentResult DataHDoesBuffer (
   DataHandler dh,
   Boolean *buffersReads,
   Boolean *buffersWrites
);
```

Parameters

dh

A data handler component.

buffersReads

A pointer to a Boolean that is returned true if the data handler component does buffer reads, false otherwise.

buffersWrites

A pointer to a Boolean that is returned true if the data handler component does buffer writes, false otherwise.

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

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`QuickTimeComponents.h`

DataHFinishData

Completes or cancels one or more queued read requests.

```
ComponentResult DataHFinishData (
   DataHandler dh,
   Ptr PlaceToPutDataPtr,
   Boolean Cancel
);
```

Parameters

`dh`

Identifies the calling program's connection to your data handler component.

`PlaceToPutDataPtr`

The location in memory that is to receive the data. The value of this parameter identifies the specific read request to be completed. If this parameter is set to `NIL`, the call affects all pending read requests.

`Cancel`

Indicates whether the calling program wants to cancel the outstanding request. If this parameter is set to TRUE, your data handler should cancel the request (or requests) identified by the `PlaceToPutDataPtr` parameter.

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

Discussion
Client programs use `DataHFinishData` either to cancel outstanding read requests or to demand that the requests be serviced immediately. Note that your component must call the client program's data-handler completion function for each queued request, even though the client program called `DataHFinishData`. Be sure to call the completion function for both canceled and completed read requests.

Special Considerations

Preroll operations are a special case of the immediate service request. The client program will have queued one or more read requests with their scheduled time of delivery set infinitely far into the future. Your data handler queues those requests until the client program calls `DataHFinishData` demanding that all outstanding read requests be satisfied immediately.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`QuickTimeComponents.h`

DataHFlushCache

Discards the contents of any cached read buffers.

```
ComponentResult DataHFlushCache (
   DataHandler dh
);
```

Parameters

dh

Identifies the calling program's connection to your data handler component.

Return Value

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

Discussion

Note that this function does not invalidate any queued read requests (made by calling your component's `DataHScheduleData` (page 60) function).

Special Considerations

Client programs may call this function if they have, in some way, changed the container associated with the current data reference on their own. Under these circumstances, data your component may have read and cached in anticipation of future read requests from the client may be invalid.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`QuickTimeComponents.h`

DataHFlushData

Forces any data in your component's write buffers to be written to the device that contains the current data reference.

```
ComponentResult DataHFlushData (
   DataHandler dh
);
```

Parameters

dh

Identifies the calling program's connection to your data handler component.

Return Value

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

Discussion

This function is essentially analogous to the Mac OS File Manager's `PBFlushFile` function. The client program may call this function after any write operation (either `DataHPutData` (page 56) or `DataHWrite` (page 71)). Your component should do what is necessary to make sure that the data is written to the storage device that contains the current data reference.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`QuickTimeComponents.h`

DataHGetAvailableFileSize

Returns the available file size for a data handler component.

```
ComponentResult DataHGetAvailableFileSize (
   DataHandler dh,
   long *fileSize
);
```

Parameters

dh

A data handler component.

fileSize

A pointer to the file size.

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

Discussion
You can call this function during an asynchronous read operation, after calling `DataHScheduleData` (page 60).

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`QuickTimeComponents.h`

DataHGetCacheSizeLimit

Returns the cache size limit for a data handler component.

```
ComponentResult DataHGetCacheSizeLimit (
   DataHandler dh,
   Size *cacheSizeLimit
);
```

Parameters

dh

A data handler component.

cacheSizeLimit

A pointer to the cache size limit.

Return Value

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

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`QuickTimeComponents.h`

DataHGetData

Reads data from its current data reference, which is a synchronous read operation.

```
ComponentResult DataHGetData (
   DataHandler dh,
   Handle h,
   long hOffset,
   long offset,
   long size
);
```

Parameters

dh

Identifies the calling program's connection to your data handler component.

h

The handle to receive the data.

hOffset

Identifies the offset into the handle where your component should return the data.

offset

The offset in the data reference from which your component is to read.

size

The number of bytes to read.

Return Value

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

Discussion

Data handler components provide two basic read facilities. `DataHGetData` function is a fully synchronous read operation, while `DataHScheduleData` (page 60) is asynchronous. (Applications provide scheduling information when they call `DataHScheduleData`.) Before any application can read data from a data reference, it must open read access to that reference by calling your component's `DataHOpenForRead` (page 50) function. `DataHCloseForRead` (page 22) closes that read access path. When your component processes the queued request, it calls the application's data-handler completion function.

Special Considerations

Note that the Movie Toolbox may try to read data from a data reference without calling your component's `DataHOpenForRead` (page 50) function. If this happens, your component should open the data reference for read-only access, respond to the read request, and then leave the data reference open in anticipation of later read requests.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`QuickTimeComponents.h`

DataHGetDataAvailability

Undocumented

```
ComponentResult DataHGetDataAvailability (
   DataHandler dh,
   long offset,
   long len,
   long *missing_offset,
   long *missing_len
);
```

Parameters

`dh`

A data handler component.

`offset`

Undocumented

`len`

Undocumented

`missing_offset`

Undocumented

`missing_len`

Undocumented

Return Value

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

Version Notes

Introduced in QuickTime 4.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`QuickTimeComponents.h`

DataHGetDataInBuffer

Returns the information about the data in a data handler component's buffer.

```
ComponentResult DataHGetDataInBuffer (
   DataHandler dh,
   long startOffset,
   long *size
);
```

Parameters

dh

A data handler component.

startOffset

The offset to the start of the data.

size

The size of the data.

Return Value

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

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`QuickTimeComponents.h`

DataHGetDataRate

Undocumented

```
ComponentResult DataHGetDataRate (
   DataHandler dh,
   long flags,
   long *bytesPerSecond
);
```

Parameters

dh

A data handler component.

flags

Undocumented

bytesPerSecond

Undocumented

Return Value

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

Version Notes

Introduced in QuickTime 5.

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

Declared In
`QuickTimeComponents.h`

DataHGetDataRef

Retrieves your component's current data reference.

```
ComponentResult DataHGetDataRef (
   DataHandler dh,
   Handle *dataRef
);
```

Parameters

dh

Identifies the calling program's connection to your data handler component.

dataRef

A pointer to a data reference handle. Your component should make a copy of its current data reference in a handle and return that handle in this field. The client program is responsible for disposing of that handle. Different types of containers may require different types of data references. For example, a reference to a memory-based movie may be a handle, while a reference to a file-based movie may be an alias. Apple's memory-based data handler for the Macintosh uses handles (and has a subtype value of `'hndl'`), while the HFS data handler uses Alias Manager aliases (its subtype value is `'alis'`). See `Data References`.

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

Discussion
All data handler components use data references to identify and locate a movie's container. Client programs can correlate data references with data handlers by matching the component's subtype value with the `data` reference type; the subtype value indicates the type of data reference the component supports. All data handlers with the same subtype value must support the same data reference type.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`QuickTimeComponents.h`

DataHGetDataRefAsType

Retrieves a data handler component's current data reference of a given type.

```
ComponentResult DataHGetDataRefAsType (
   DataHandler dh,
   OSType requestedType,
   Handle *dataRef
);
```

Parameters

dh

A data handler component.

requestedType

The type of the data reference to retrieve; see `Data References`.

dataRef

A pointer to a data reference handle. Your component should make a copy of its current data reference in a handle and return that handle in this field. The client program is responsible for disposing of that handle. Different types of containers may require different types of data references. For example, a reference to a memory-based movie may be a handle, while a reference to a file-based movie may be an alias. Apple's memory-based data handler for the Macintosh uses handles (and has a subtype value of `'hndl'`), while the HFS data handler uses Alias Manager aliases (its subtype value is `'alis'`). See `Data References`.

Return Value

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

Version Notes

Introduced in QuickTime 4.1.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`QuickTimeComponents.h`

DataHGetDataRefExtension

Retrieves your component's current data reference extension data.

```
ComponentResult DataHGetDataRefExtension (
   DataHandler dh,
   Handle *extension,
   OSType idType
);
```

Parameters

dh

A data handler component.

extension

A pointer to a handle to the extension data.

idType

A four-byte signature identifying the type of extension data.

Return Value

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

Version Notes
Introduced in QuickTime 4.1.

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

Declared In
`QuickTimeComponents.h`

DataHGetDataRefWithAnchor

Retrieves a data handler component's component's current data reference and anchor data reference.

```
ComponentResult DataHGetDataRefWithAnchor (
   DataHandler dh,
   Handle anchorDataRef,
   OSType dataRefType,
   Handle *dataRef
);
```

Parameters

dh

A data handler component.

anchorDataRef

A handle to the anchor data reference.

dataRefType

The type of the data reference; see `Data References`.

dataRef

A pointer to a data reference handle. Your component should make a copy of its current data reference in a handle and return that handle in this field. The client program is responsible for disposing of that handle. Different types of containers may require different types of data references. For example, a reference to a memory-based movie may be a handle, while a reference to a file-based movie may be an alias. Apple's memory-based data handler for the Macintosh uses handles (and has a subtype value of `'hndl'`), while the HFS data handler uses Alias Manager aliases (its subtype value is `'alis'`). See `Data References`.

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

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`QuickTimeComponents.h`

DataHGetDeviceIndex

Returns a value that identifies the device on which a data reference resides.

```
ComponentResult DataHGetDeviceIndex (
   DataHandler dh,
   long *deviceIndex
);
```

Parameters

dh

Identifies the calling program's connection to your data handler component.

deviceIndex

A pointer to a field that your data handler component uses to return a device identifier value. Your component may use any identifier value that is appropriate (for example, Apple's HFS data handler uses the volume reference number). The client program should do nothing with this value other than compare it with other identifiers returned by your data handler component.

Return Value

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

Discussion

Some client programs may need to account for the fact that two or more data references reside on the same device. For instance, this may affect storage-allocation requirements. This function allows such client programs to obtain this information from your data handler.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`QuickTimeComponents.h`

DataHGetFileName

Retrieves the name of the file supplying the current data reference for a data handler.

```
ComponentResult DataHGetFileName (
   DataHandler dh,
   Str255 str
);
```

Parameters

dh

A data handler component.

str

The name of the file as a string.

Return Value

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

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

DataHGetFileSize

Returns the size, in bytes, of the current data reference.

```
ComponentResult DataHGetFileSize (
   DataHandler dh,
   long *fileSize
);
```

Parameters

`dh`

Identifies the calling program's connection to your data handler component.

`fileSize`

A pointer to a long integer. Your component returns the size of the container corresponding to the current data reference, in bytes.

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

Discussion
This function is operationally equivalent to the Mac OS File Manager's `GetEOF` function. Before writing data to a data reference, applications must call your component's `DataHOpenForWrite` (page 51) function to open a write path to the container. `DataHCloseForWrite` (page 23) closes that write path. As is the case with `DataHScheduleData` (page 60), your component calls the application's data-handler completion function when you are done with the write request.

Special Considerations

Note that some data handlers may not support write operations. For example, some shared devices, such as a CD-ROM "jukebox," may be read-only devices. As a result, it is very important that your data handler correctly report its write capabilities to client programs.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
ElectricImageComponent.win

qtdataref

qtdataref.win

ThreadsImporter

ThreadsImportMovie

Declared In
`QuickTimeComponents.h`

DataHGetFileSize64

Provides a 64-bit version of DataHGetFileSize.

```
ComponentResult DataHGetFileSize64 (
   DataHandler dh,
   wide *fileSize
);
```

Parameters

dh

Identifies the calling program's connection to your data handler component.

fileSize

A pointer to a wide. Your component returns the size of the container corresponding to the current data reference, in bytes.

Return Value

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

Discussion

The only difference between this function and `DataHGetFileSize` (page 38) is that the `fileSize` parameter is a 64-bit integer instead of a 32-bit integer.

Special Considerations

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

Version Notes

Introduced in QuickTime 4.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`QuickTimeComponents.h`

DataHGetFileSizeAsync

Returns the size of the current data reference, invoking a completion callback.

```
ComponentResult DataHGetFileSizeAsync (
   DataHandler dh,
   wide *fileSize,
   DataHCompletionUPP completionRtn,
   long refCon
);
```

Parameters

dh

Identifies the calling program's connection to your data handler component.

fileSize

A pointer to a 64-bit value. Your component returns the size of the container corresponding to the current data reference, in bytes.

completionRtn

A pointer to a data handler completion callback, described in `DataHCompletionProc`.

refCon

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

Return Value

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

Version Notes

Introduced in QuickTime 4.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`QuickTimeComponents.h`

DataHGetFileTypeOrdering

Returns the preferred ordering for file typing information.

```
ComponentResult DataHGetFileTypeOrdering (
   DataHandler dh,
   DataHFileTypeOrderingHandle *orderingListHandle
);
```

Parameters

dh

Identifies the calling program's connection to your data handler component.

orderingListHandle

A pointer to a handle to a list of `OSType` values (see below). The list may contain only a subset of the currently defined types (i.e., Mac OS file type, extension, MIME type) to limit the consideration to reasonable types. See these constants:

`kDataHFileTypeMacOSFileType`

`kDataHFileTypeExtension`

`kDataHFileTypeMIME`

Return Value

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

Discussion

If the data handler has not set the data reference, it can choose to return either an error or a reasonable default ordering list.

Special Considerations

Before making a call to this function, the client should have opened the data handler and called `DataHSetDataRef` (page 63) or `DataHSetDataRefWithAnchor` (page 65). This allows the data handler to return a different ordering based on the particular file. This might allow for a data handler to vary its ordering based on the location of the file. For example, on the Mac OS, it might use extensions only on foreign volumes. For other volumes, it might use a Mac OS file type followed by a file extension.

Version Notes

Introduced in QuickTime 5.

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

Declared In
`QuickTimeComponents.h`

DataHGetFreeSpace

Reports the number of bytes available on the device that contains the current data reference.

```
ComponentResult DataHGetFreeSpace (
   DataHandler dh,
   unsigned long *freeSize
);
```

Parameters

dh

Identifies the calling program's connection to your data handler component.

freeSize

A pointer to an unsigned long integer. Your component returns the number of bytes of free space available on the device that contains the container referred to by the current data reference.

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

Discussion
Before writing data to a data reference, applications must call your component's `DataHOpenForWrite` (page 51) function to open a write path to the container. `DataHCloseForWrite` (page 23) closes that write path. As is the case with `DataHScheduleData` (page 60), your component calls the application's data-handler completion function when you are done with the write request.

Special Considerations

Note that some data handlers may not support write operations. For example, some shared devices, such as a CD-ROM "jukebox," may be read-only devices. As a result, it is very important that your data handler correctly report its write capabilities to client programs.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`QuickTimeComponents.h`

DataHGetFreeSpace64

Provides a 64-bit version of DataHGetFreeSpace.

```
ComponentResult DataHGetFreeSpace64 (
   DataHandler dh,
   wide *freeSize
);
```

Parameters

dh

A data handler component.

freeSize

A pointer to a 64-bit integer. Your component returns the number of bytes of free space available on the device that contains the container referred to by the current data reference.

Return Value

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

Discussion

The only difference between this function and `DataHGetFreeSpace` (page 41) is that the `freeSize` parameter is a 64-bit integer instead of a 32-bit integer.

Special Considerations

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

Version Notes

Introduced in QuickTime 4.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`QuickTimeComponents.h`

DataHGetInfo

Retrieves information from a data handler.

```
ComponentResult DataHGetInfo (
   DataHandler dh,
   OSType what,
   void *info
);
```

Parameters

dh

A data handler component.

what

A selector for the information requested. No selectors are currently defined.

info

A pointer to an area of memory in which to place the retrieved information.

Return Value

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

Version Notes

Introduced in QuickTime 6.

Availability
Available in Mac OS X v10.1 and later.

Declared In
`QuickTimeComponents.h`

DataHGetInfoFlags

Provides information about the operation of a data handler component.

```
ComponentResult DataHGetInfoFlags (
   DataHandler dh,
   UInt32 *flags
);
```

Parameters

dh

A data handler component.

flags

Flags (see below) that provide information about the data handler. See these constants:

`kDataHInfoFlagNeverStreams`

`kDataHInfoFlagCanUpdateDataRefs`

`kDataHInfoFlagNeedsNetworkBandwidth`

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

Version Notes
Introduced in QuickTime 4.

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

Related Sample Code
Fiendishthngs

Declared In
`QuickTimeComponents.h`

DataHGetMacOSFileType

Gets the Mac OS file type for a data handler's current data reference.

```
ComponentResult DataHGetMacOSFileType (
   DataHandler dh,
   OSType *fileType
);
```

Parameters

dh

A data handler component.

fileType

A pointer to a file type; see `File Types and Creators`.

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

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
ThreadsImportMovie

Declared In
`QuickTimeComponents.h`

DataHGetMIMEType

Gets the MIME type for a data handler's current data reference.

```
ComponentResult DataHGetMIMEType (
   DataHandler dh,
   Str255 mimeType
);
```

Parameters

dh

A data handler component.

mimeType

A MIME type string.

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

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
ThreadsImportMovie

Declared In
`QuickTimeComponents.h`

DataHGetMIMETypeAsync

Performs asynchronous discovery of a HTTP/FTP connection's MIME type.

```
ComponentResult DataHGetMIMETypeAsync (
   DataHandler dh,
   Str255 mimeType,
   DataHCompletionUPP completionRtn,
   long refCon
);
```

Parameters

dh

A data handler component.

mimeType

The MIME type string when (and if) it becomes available.

completionRtn

A `DataHCompletionProc` callback that is called when either the data becomes available or there is a failure. Failure can happen if there is a timeout or `DataHFinishData` (page 28) is called with a cancel instruction. The `mimeType` parameter will not be updated until the complete routine executes. If a completion routine is not specified, the call will return immediately.

refCon

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

Return Value

If the MIME type is known, this function updates `mimeType` and returns `noErr`. If the information is not known yet, the error `notEnoughDataErr` is returned. This allows non-blocking calls to be made to this function. If it returns another error, that indicates some other failure; see `Error Codes`.

Discussion

This function removes synchronous blocks from QuickTime's movie opening code.

Version Notes

Introduced in QuickTime 5.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`QuickTimeComponents.h`

DataHGetMovie

Gets the movie for a data handler's current data reference.

```
ComponentResult DataHGetMovie (
   DataHandler dh,
   Movie *theMovie,
   short *id
);
```

Parameters

dh

A data handler component.

theMovie

A movie identifier. This is the same as the identifier your application obtains from such functions as `NewMovie`, `NewMovieFromFile`, and `NewMovieFromHandle`.

id

A pointer to the field that specifies the resource containing the movie data.

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

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`QuickTimeComponents.h`

DataHGetMovieWithFlags

Gets the movie for a data handler's current data reference, allowing the flags that would be passed to NewMovieFromDataRef to be passed to the handler.

```
ComponentResult DataHGetMovieWithFlags (
   DataHandler dh,
   Movie *theMovie,
   short *id,
   short flags
);
```

Parameters

dh

A data handler component.

theMovie

A movie identifier. This is the same as the identifier your application obtains from such functions as `NewMovie`, `NewMovieFromFile`, and `NewMovieFromHandle`.

id

A pointer to the field that specifies the resource containing the movie data.

flags

Flags (see below) that control movie characteristics. See these constants:

`newMovieActive`

`newMovieDontResolveDataRefs`

`newMovieDontAskUnresolvedDataRefs`

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

Discussion
Passing these flags lets you control whether or not the movie returned is active. Additionally, it allows `async` movie loading to be more efficient.

Version Notes
Introduced in QuickTime 4.1.

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

Declared In
`QuickTimeComponents.h`

DataHGetPreferredBlockSize

Reports the block size that it prefers to use when accessing the current data reference.

```
ComponentResult DataHGetPreferredBlockSize (
   DataHandler dh,
   long *blockSize
);
```

Parameters

dh

Identifies the calling program's connection to your data handler component.

blockSize

A pointer to a long integer. Your component returns the size of blocks (in bytes) it prefers to use when accessing the current data reference.

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

Discussion
Different devices use different file system block sizes. This function allows your component to report its preferred block size to the client program. Note that the client program is not required to use this block size when making requests. Some clients may, however, try to accommodate your component's preference. Applications may call your component's `DataHGetPreferredBlockSize` function in order to determine how best to interact with your data handler. As is the case with `DataHScheduleData` (page 60), your component calls the application's data-handler completion function when you are done with the write request.

Special Considerations

Note that some data handlers may not support write operations. For example, some shared devices, such as a CD-ROM "jukebox," may be read-only devices. As a result, it is very important that your data handler correctly report its write capabilities to client programs.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`QuickTimeComponents.h`

DataHGetScheduleAheadTime

Reports how far in advance it prefers clients to issue read requests.

```
ComponentResult DataHGetScheduleAheadTime (
   DataHandler dh,
   long *millisecs
);
```

Parameters

dh

Identifies the calling program's connection to your data handler component.

millisecs

A pointer to a long integer. Your component should set this field with a value indicating the number of milliseconds you prefer to receive read requests in advance of the time when the data must be delivered.

Return Value

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

Discussion

This function allows your data handler to tell the client program how far in advance it should schedule its read requests. By default, the Movie Toolbox issues scheduled read requests between 1 and 2 seconds before it needs the data from those requests. For some data handlers, however, this may not be enough time. For example, some data handlers may have to accommodate network delays when processing read requests. Client programs that call `DataHGetScheduleAheadTime` may try to respect your component's preference.

Special Considerations

Note that not all client programs will call `DataHGetScheduleAheadTime`. Further, some clients may not be able to accommodate your preferred time in all cases, even if they have asked for your component's preference. As a result, your component should have a strategy for handling requests that don't provide enough advanced scheduling time. For example, if your component receives a `DataHScheduleData` (page 60) request that it cannot satisfy, it can fail the request with an appropriate error code.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`QuickTimeComponents.h`

DataHGetTemporaryDataRefCapabilities

Undocumented

```
ComponentResult DataHGetTemporaryDataRefCapabilities (
   DataHandler dh,
   long *outUnderstoodFlags
);
```

Parameters

dh

A data handler component.

`outUnderstoodFlags`

Undocumented

Return Value

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

Version Notes

Introduced in QuickTime 6.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`QuickTimeComponents.h`

DataHGetVolumeList

Returns a list of the volumes your component can access, along with flags indicating your component's capabilities for each volume.

```
ComponentResult DataHGetVolumeList (
   DataHandler dh,
   DataHVolumeList *volumeList
);
```

Parameters

`dh`

Identifies the calling program's connection to your data handler component.

`volumeList`

A pointer to a field that your data handler component uses to return a handle to a volume list. Your component constructs the volume list by allocating a handle and filling it with a series of `DataHVolumeListRecord` structures, one structure for each volume your component can access.

Return Value

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

Discussion

To reduce the delay that may result from choosing an appropriate data handler for a volume, the Movie Toolbox maintains a list of data handlers and the volumes they support. The Movie Toolbox uses `DataHGetVolumeList` to build that list. Your data handler may use any facilities necessary to determine whether it can access the volume, including opening a container on the volume. Your component should set to 1 as many of the capability flags as are appropriate for each volume. Don't include records for volumes your handler cannot support. For example, if your component supports networked multimedia servers using a special set of protocols, your data handler should set the `kDataHCanRead` and `kDataHCanSpecialRead` flags to 1 for any volume that is on that server. In addition, if your component can write to a volume on the server, set the `kDataHCanWrite` and `kDataHCanSpecialWrite` flags to 1 (perhaps along with `kDataHCanStreamingWrite`). However, your component should create entries only for those volumes that support your protocols.

Special Considerations

It is the calling program's responsibility to dispose of the handle returned by your component. The Movie Toolbox tracks mounting and unmounting removable volumes, and keeps its volume list current. As a result, the Movie Toolbox may call your component's `DataHGetVolumeList` function whenever a removable volume is mounted. If your data handler does not process data that is stored in file system volumes, you need not support this function.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`QuickTimeComponents.h`

DataHIsStreamingDataHandler

Determines if a data handler handles streaming data.

```
ComponentResult DataHIsStreamingDataHandler (
   DataHandler dh,
   Boolean *yes
);
```

Parameters

dh

A data handler component.

yes

Pointer to a Boolean that returns TRUE if the data handler handles streaming data, FALSE otherwise.

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

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`QuickTimeComponents.h`

DataHOpenForRead

Opens a data handler's current data reference for read-only access.

```
ComponentResult DataHOpenForRead (
   DataHandler dh
);
```

Parameters

dh

Identifies the calling program's connection to your data handler component.

Return Value

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

Discussion

After setting your component's current data reference by calling `DataHSetDataRef` (page 63), client programs call `DataHOpenForRead` to start reading from the data reference. Your component should open the data reference for read-only access. If the data reference is already open or cannot be opened, return an appropriate error code. As is the case with `DataHScheduleData` (page 60), your component calls the application's data-handler completion function when you are done with the write request.

Special Considerations

Note that the Movie Toolbox may try to read data from a data reference without calling your component's `DataHOpenForRead` function. If this happens, your component should open the data reference for read-only access, respond to the read request, and then leave the data reference open in anticipation of later read requests.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

ElectricImageComponent.win

qtdataref

qtdataref.win

ThreadsImporter

ThreadsImportMovie

Declared In

`QuickTimeComponents.h`

DataHOpenForWrite

Opens your component's current data reference for write-only access.

```
ComponentResult DataHOpenForWrite (
   DataHandler dh
);
```

Parameters

dh

Identifies the calling program's connection to your data handler component.

Return Value

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

Discussion

After setting your component's current data reference by calling `DataHSetDataRef` (page 63), client programs call `DataHOpenForWrite` to start writing to the data reference. Your component should open the data reference for write-only access. If the data reference is already open or cannot be opened, return an appropriate error code. As is the case with `DataHScheduleData` (page 60), your component calls the application's data-handler completion function when you are done with the write request.

Special Considerations

Note that some data handlers may not support write operations. For example, some shared devices, such as a CD-ROM "jukebox," may be read-only devices. As a result, it is very important that your data handler correctly report its write capabilities to client programs.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

ElectricImageComponent.win

qtdataref

qtdataref.win

ThreadsImporter

ThreadsImportMovie

Declared In

`QuickTimeComponents.h`

DataHPlaybackHints

Provides additional information to your component that you may use to optimize the operation of your data handler.

```
ComponentResult DataHPlaybackHints (
   DataHandler dh,
   long flags,
   unsigned long minFileOffset,
   unsigned long maxFileOffset,
   long bytesPerSecond
);
```

Parameters

dh

Identifies the calling program's connection to your data handler component.

flags

Reserved. Set this parameter to 0.

minFileOffset

Together with the `maxFileOffset` parameter, specifies the range of data the client program anticipates using from the current data reference. This parameter specifies the offset of earliest byte the program expects to use (that is, the minimum container offset value). If the client expects to access bytes from the beginning of the container, it should set this parameter to 0.

maxFileOffset

The offset of the latest byte the program expects to use (that is, the maximum container offset value). If the client expects to use bytes throughout the container, the client should set this parameter to -1.

bytesPerSecond

The rate in bytes per second at which your data handler must read data from the data reference in order to keep up with the client program's anticipated needs.

Return Value

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

Discussion

Applications may call your handler's `DataHPlaybackHints` function to provide you with some guidelines about how those applications plan to use the current data reference. Your component should be prepared to have this function called more than once for a given data reference. For example, the Movie Toolbox calls this function whenever a movie's playback rate changes. This is a handy way for your data handler to track playback rate changes.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`QuickTimeComponents.h`

DataHPlaybackHints64

Provides a 64-bit version of DataHPlaybackHints.

```
ComponentResult DataHPlaybackHints64 (
   DataHandler dh,
   long flags,
   const wide *minFileOffset,
   const wide *maxFileOffset,
   long bytesPerSecond
);
```

Parameters

dh

Identifies the calling program's connection to your data handler component.

flags

Reserved. Set this parameter to 0.

minFileOffset

Together with the `maxFileOffset` parameter, specifies the range of data the client program anticipates using from the current data reference. This parameter points to the offset of the earliest byte the program expects to use (that is, the minimum container offset value). If the client expects to access bytes from the beginning of the container, it should set this parameter to 0.

maxFileOffset

Pointer to the offset of the latest byte the program expects to use (that is, the maximum container offset value). If the client expects to use bytes throughout the container, the client should set this parameter to -1.

bytesPerSecond

The rate in bytes per second at which your data handler must read data from the data reference in order to keep up with the client program's anticipated needs.

Return Value

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

Discussion

The only difference between this function and `DataHPlaybackHints` (page 52) is that the `minFileOffset` parameter and `maxFileOffset` parameter are 64-bit integers instead of 32-bit integers.

Special Considerations

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

Version Notes

Introduced in QuickTime 4.1.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`QuickTimeComponents.h`

DataHPollRead

Undocumented

```
ComponentResult DataHPollRead (
   DataHandler dh,
   void *dataPtr,
   UInt32 *dataSizeSoFar
);
```

Parameters

dh

A data handler component.

dataPtr

Undocumented

dataSizeSoFar

Undocumented

Return Value

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

Version Notes

Introduced in QuickTime 4.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`QuickTimeComponents.h`

DataHPreextend

Allocates new space for the current data reference, enlarging the container.

```
ComponentResult DataHPreextend (
   DataHandler dh,
   unsigned long maxToAdd,
   unsigned long *spaceAdded
);
```

Parameters

dh

Identifies the calling program's connection to your data handler component.

maxToAdd

The amount of space to add to the current data reference, in bytes. If the client program sets this parameter to 0, your component should add as much space as it can.

spaceAdded

A pointer to an unsigned long integer. Your component returns the number of bytes it was able to add to the data reference, in bytes.

Return Value

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

Discussion

This function asks your component to make a container larger, analogous to the Mac OS File Manager's `PBAllocContig` function. When it is called, your component should allocate contiguous free space. If there is not sufficient contiguous free space to satisfy the request, your component should return a `dskFulErr` error code. Client programs use this function in order to avoid incurring any space-allocation delay when capturing movie data. As is the case with `DataHScheduleData` (page 60), your component calls the application's data-handler completion function when you are done with the write request.

Special Considerations

Note that some data handlers may not support write operations. For example, some shared devices, such as a CD-ROM "jukebox," may be read-only devices. As a result, it is very important that your data handler correctly report its write capabilities to client programs.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`QuickTimeComponents.h`

DataHPreextend64

Provides a 64-bit version of DataHPreextend.

```
ComponentResult DataHPreextend64 (
   DataHandler dh,
   const wide *maxToAdd,
   wide *spaceAdded
);
```

Parameters

dh

Identifies the calling program's connection to your data handler component.

maxToAdd

The amount of space to add to the current data reference, in bytes. If the client program sets this parameter to 0, your component should add as much space as it can.

spaceAdded

A pointer to a 64-bit integer. Your component returns the number of bytes it was able to add to the data reference, in bytes.

Return Value

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

Discussion

The only difference between this function and `DataHPreextend` (page 55) is that the `spaceAdded` parameter is a 64-bit integer instead of a 32-bit integer.

Special Considerations

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

Version Notes

Introduced in QuickTime 4.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`QuickTimeComponents.h`

DataHPutData

Writes data to a component's current data reference.

```
ComponentResult DataHPutData (
   DataHandler dh,
   Handle h,
   long hOffset,
   long *offset,
   long size
);
```

Parameters

dh

Identifies the calling program's connection to your data handler component.

h

The handle that contains the data to be written to the data reference.

hOffset

Identifies the offset into the handle h to the data to be written.

offset

A pointer to a long integer. Your component returns the offset in the data reference at which your component wrote the data.

size

The number of bytes to write.

Return Value

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

Discussion

This function provides a high-level write interface. This is a synchronous write operation that only appends data to the end of the current data reference. That is, the client program's execution is blocked until your component returns control from this function, and the client cannot control where the data is written. As a result, most movie-capture clients (for example, Apple's sequence grabber component) use `DataHWrite` (page 71) to write data when creating movies. As is the case with `DataHScheduleData` (page 60), your component calls the application's data-handler completion function when you are done with the write request.

Special Considerations

Note that some data handlers may not support write operations. For example, some shared devices, such as a CD-ROM "jukebox," may be read-only devices. As a result, it is very important that your data handler correctly report its write capabilities to client programs.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`QuickTimeComponents.h`

DataHReadAsync

Undocumented

```
ComponentResult DataHReadAsync (
   DataHandler dh,
   void *dataPtr,
   UInt32 dataSize,
   const wide *dataOffset,
   DataHCompletionUPP completion,
   long refCon
);
```

Parameters

dh

A data handler component.

dataPtr

Undocumented

dataSize

Undocumented

`dataOffset`

Undocumented

`completion`

A pointer to a data-handler completion callback, described in `DataHCompletionProc`.

`refCon`

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

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

Version Notes
Introduced in QuickTime 4.

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

Related Sample Code
qtdataref

qtdataref.win

qtfiletransfer

ThreadsImporter

ThreadsImportMovie

Declared In
`QuickTimeComponents.h`

DataHRenameFile

Undocumented

```
ComponentResult DataHRenameFile (
   DataHandler dh,
   Handle newDataRef
);
```

Parameters

`dh`

A data handler component.

`newDataRef`

A handle to a new QuickTime data reference.

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

Version Notes
Introduced in QuickTime 6.

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

Declared In
`QuickTimeComponents.h`

DataHResolveDataRef

Locates the container associated with a given data reference.

```
ComponentResult DataHResolveDataRef (
   DataHandler dh,
   Handle theDataRef,
   Boolean *wasChanged,
   Boolean userInterfaceAllowed
);
```

Parameters

dh

Identifies the calling program's connection to your data handler component.

theDataRef

The data reference to be resolved. Different types of containers may require different types of data references. For example, a reference to a memory-based movie may be a handle, while a reference to a file-based movie may be an alias. Apple's memory-based data handler for the Macintosh uses handles (and has a subtype value of `'hndl'`), while the HFS data handler uses Alias Manager aliases (its subtype value is `'alis'`). See `Data References`.

wasChanged

A pointer to a Boolean. Your component should set that Boolean to TRUE if, in locating the container, your data handler updates any information in the data reference.

userInterfaceAllowed

Indicates whether your component may interact with the user when locating the `container`. If this parameter is set to TRUE, your component may ask the user to help locate the container (for instance, by presenting a Find File dialog box).

Return Value

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

Discussion

This function permits your component to locate a data reference's container. This function is equivalent to the Mac OS Alias Manager's `ResolveAlias` function. The client program asks your component to locate the container that is associated with a given data reference. If your component determines that the data reference needs to be updated with more accurate location information, it should put the new information in the supplied data reference (and set the Boolean referred to by the `wasChanged` parameter to TRUE). Client programs can correlate data references with data handlers by matching the component's subtype value with the `data` reference type; the subtype value indicates the type of data reference the component supports. All data handlers with the same subtype value must support the same data reference type.

Special Considerations

Client programs may call your data handler's `DataHResolveDataRef` function at any time. Typically, however, the Movie Toolbox uses this function as part of its strategy for opening and reading a movie container. As such, you can expect that the supplied data reference will identify a container that your component can support.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

DataHScheduleData

Reads data from its current data reference, which can be a synchronous read operation or an asynchronous read operation.

```
ComponentResult DataHScheduleData (
   DataHandler dh,
   Ptr PlaceToPutDataPtr,
   long FileOffset,
   long DataSize,
   long RefCon,
   DataHSchedulePtr scheduleRec,
   DataHCompletionUPP CompletionRtn
);
```

Parameters

dh
Identifies the calling program's connection to your data handler component.

PlaceToPutDataPtr
The location in memory that is to receive the data.

FileOffset
The offset in the data reference from which your component is to read.

DataSize
The number of bytes to read.

RefCon
A reference constant that your data handler component should provide to the data-handler completion function specified with the `CompletionRtn` parameter.

scheduleRec
A pointer to a `DataHScheduleRecord`. If this parameter is set to `NIL`, then the client program is requesting a synchronous read operation (that is, your data handler must return the data before returning control to the client program). If this parameter is not set to `NIL`, it must contain the location of a schedule record that has timing information for an asynchronous read request. Your data handler should return control to the client program immediately, and then call the client's data-handler completion function when the data is ready.

CompletionRtn
A pointer to a data handler completion function, described in `DataHCompletionProc`. The client program must provide a completion routine for all asynchronous read requests (that is, all requests that include a valid schedule record). For synchronous requests, client programs should set this parameter to `NIL`. However, if the function is provided, your handler must call it, even after synchronous requests. When your data handler finishes with the client program's read request, your component must call this routine even if the request fails. Your component should pass the reference constant that the client program provided with the `RefCon` parameter.

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

Discussion

This function provides both a synchronous and an asynchronous read interface. Synchronous read operations work like the `DataHGetData` (page 31) function; the data handler component returns control to the client program only after it has serviced the read request. Asynchronous read operations allow client programs to schedule read requests in the context of a specified QuickTime time base. Your data handler queues the request and immediately returns control to the calling program. After your component actually reads the data, it calls the client program's data-handler completion function. If your component cannot satisfy the request (for example, the request requires data more quickly than you can deliver it), your component should reject the request immediately, rather than queuing the request and then calling the client's data-handler completion function.

```
typedef struct DataHScheduleRecord {
                    TimeRecord timeNeededBy; /* schedule info */
                    long     extendedID; /* type of data */
                    long     extendedVers; /* reserved */
                    Fixed   priority;  /* priority */
                    } DataHScheduleRecord, *DataHSchedulePtr;
```

Special Considerations

Note that the Movie Toolbox may try to read data from a data reference without calling your component's `DataHOpenForRead` (page 50) function. If this happens, your component should open the data reference for read-only access, respond to the read request, and then leave the data reference open in anticipation of later read requests.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

ElectricImageComponent

ElectricImageComponent.win

Declared In

`QuickTimeComponents.h`

DataHScheduleData64

Provides a 64-bit version of DataHScheduleData.

```
ComponentResult DataHScheduleData64 (
   DataHandler dh,
   Ptr PlaceToPutDataPtr,
   const wide *FileOffset,
   long DataSize,
   long RefCon,
   DataHSchedulePtr scheduleRec,
   DataHCompletionUPP CompletionRtn
);
```

Parameters

dh

Identifies the calling program's connection to your data handler component.

PlaceToPutDataPtr
: The location in memory that is to receive the data.

FileOffset
: A pointer to the offset in the data reference from which your component is to read.

DataSize
: The number of bytes to read.

RefCon
: A reference constant that your data handler component should provide to the data-handler completion function specified with the `CompletionRtn` parameter.

scheduleRec
: A pointer to a `DataHScheduleRecord`. If this parameter is set to `NIL`, then the client program is requesting a synchronous read operation (that is, your data handler must return the data before returning control to the client program). If this parameter is not set to `NIL`, it must contain the location of a schedule record that has timing information for an asynchronous read request. Your data handler should return control to the client program immediately, and then call the client's data-handler completion function when the data is ready.

CompletionRtn
: A pointer to a data handler completion function, described in `DataHCompletionProc`. The client program must provide a completion routine for all asynchronous read requests (that is, all requests that include a valid schedule record). For synchronous requests, client programs should set this parameter to `NIL`. However, if the function is provided, your handler must call it, even after synchronous requests. When your data handler finishes with the client program's read request, your component must call this routine even if the request fails. Your component should pass the reference constant that the client program provided with the `RefCon` parameter.

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

Discussion
The only difference between this function and `DataHScheduleData` (page 60) is that the `FileOffset` parameter is a 64-bit integer instead of a 32-bit integer.

Special Considerations

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

Version Notes
Introduced in QuickTime 4.

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

Declared In
`QuickTimeComponents.h`

DataHSetCacheSizeLimit

Sets the cache size limit for a data handler component.

```
ComponentResult DataHSetCacheSizeLimit (
   DataHandler dh,
   Size cacheSizeLimit
);
```

Parameters

dh

A data handler component.

cacheSizeLimit

A pointer to the cache size limit.

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

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`QuickTimeComponents.h`

DataHSetDataRef

Assigns a data reference to your data handler component.

```
ComponentResult DataHSetDataRef (
   DataHandler dh,
   Handle dataRef
);
```

Parameters

dh

Identifies the calling program's connection to your data handler component.

dataRef

The data reference. This parameter contains a handle to the information that identifies the container in question. Different types of containers may require different types of data references. For example, a reference to a memory-based movie may be a handle, while a reference to a file-based movie may be an alias. For example, Apple's memory-based data handler for the Macintosh uses handles (and has a subtype value of `'hndl'`), while the HFS data handler uses Alias Manager aliases (its subtype value is `'alis'`). The client program is responsible for disposing of the handle, so your component must make a copy of it. See `Data References`.

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

Discussion
This function allows your application to assign your data handler's current data reference. All data handler components use data references to identify and locate a movie's container. Client programs can correlate data references with data handlers by matching the component's subtype value with the `data` reference type; the subtype value indicates the type of data reference the component supports. All data handlers with the same subtype value must support the same data reference type.

Special Considerations

Note that the type of data reference always corresponds to the type that your component supports, and that you specify in the `component` subtype value of your data handler. As a result, the client program does not provide a data reference type value (unlike the Movie Toolbox's data reference functions).

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
ElectricImageComponent.win

qtdataref

qtdataref.win

ThreadsImporter

ThreadsImportMovie

Declared In
`QuickTimeComponents.h`

DataHSetDataRefExtension

Sets your component's current data reference extension data.

```
ComponentResult DataHSetDataRefExtension (
   DataHandler dh,
   Handle extension,
   OSType idType
);
```

Parameters

`dh`

A data handler component.

`extension`

A handle to the extension data.

`idType`

A four-byte signature identifying the type of extension data.

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

Version Notes
Introduced in QuickTime 4.1.

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

Declared In
`QuickTimeComponents.h`

DataHSetDataRefWithAnchor

Sets the data reference and anchor data reference for a data handler.

```
ComponentResult DataHSetDataRefWithAnchor (
   DataHandler dh,
   Handle anchorDataRef,
   OSType dataRefType,
   Handle dataRef
);
```

Parameters

dh

A data handler component.

anchorDataRef

A handle to the anchor data reference.

dataRefType

The type of the data reference. Different types of containers may require different types of data references. For example, a reference to a memory-based movie may be a handle, while a reference to a file-based movie may be an alias. Apple's memory-based data handler for the Macintosh uses handles (and has a subtype value of `'hndl'`), while the HFS data handler uses Alias Manager aliases (its subtype value is `'alis'`). See `Data References`.

dataRef

A data reference handle. Your component should make a copy of its current data reference in a handle and return that handle in this field. The client program is responsible for disposing of that handle.

Return Value

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

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`QuickTimeComponents.h`

DataHSetFileSize

Sets the size, in bytes, of the current data reference.

```
ComponentResult DataHSetFileSize (
   DataHandler dh,
   long fileSize
);
```

Parameters

dh

Identifies the calling program's connection to your data handler component.

fileSize

The new size of the container corresponding to the current data reference, in bytes.

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

Discussion
This function is operationally equivalent to the Mac OS File Manager's `SetEOF` function. If the client program specifies a new size that is greater than the current size, your component should extend the container to accommodate that new size. If the client program specifies a container size of 0, your component should free all of the space occupied by the container.

Special Considerations

Note that some data handlers may not support write operations. For example, some shared devices, such as a CD-ROM "jukebox," may be read-only devices. As a result, it is very important that your data handler correctly report its write capabilities to client programs.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Declared In
`QuickTimeComponents.h`

DataHSetFileSize64

Provides a 64-bit version of DataHSetFileSize.

```
ComponentResult DataHSetFileSize64 (
   DataHandler dh,
   const wide *fileSize
);
```

Parameters

`dh`

Identifies the calling program's connection to your data handler component.

`fileSize`

A pointer to the new size of the container corresponding to the current data reference, in bytes.

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

Discussion
The only difference between this function and `DataHSetFileSize` (page 65) is that the `fileSize` parameter is a 64-bit integer instead of a 32-bit integer.

Special Considerations

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

Version Notes
Introduced in QuickTime 4.

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

Declared In
`QuickTimeComponents.h`

DataHSetIdleManager

Lets a data handler report its idling needs.

```
ComponentResult DataHSetIdleManager (
   DataHandler dh,
   IdleManager im
);
```

Parameters

dh

A data handler component.

im

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

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

Discussion
This routine must be implemented by a data handler if the handler needs to report its idling requirements. If you have a handler that supports scheduling reads in the future, you can schedule calls to `DataHTask` (page 70) via the data structure returned by this function.

Version Notes
Introduced in QuickTime 6.

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

Declared In
`QuickTimeComponents.h`

DataHSetMacOSFileType

Sets the Mac OS file type for a data handler's current data reference.

```
ComponentResult DataHSetMacOSFileType (
   DataHandler dh,
   OSType fileType
);
```

Parameters

dh

A data handler component.

fileType

A file type; see `File Types and Creators`.

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

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code
ElectricImageComponent

ElectricImageComponent.win

Declared In
`QuickTimeComponents.h`

DataHSetMovieUsageFlags

Sets the way that a data handler appends data to its storage.

```
ComponentResult DataHSetMovieUsageFlags (
   DataHandler dh,
   long flags
);
```

Parameters

dh

A data handler component.

flags

Constants (see below) that control data appending. See these constants:

`kDataHMovieUsageDoAppendMDAT`

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

Version Notes
Introduced in QuickTime 6.

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

Declared In
`QuickTimeComponents.h`

DataHSetTimeBase

Sets the time base for a data handler component.

```
ComponentResult DataHSetTimeBase (
   DataHandler dh,
   TimeBase tb
);
```

Parameters

dh

A data handler component.

tb

A pointer to a `TimeBaseRecord` structure.

Return Value

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

Version Notes

Introduced in QuickTime 4.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`QuickTimeComponents.h`

DataHSetTimeHints

Undocumented

```
ComponentResult DataHSetTimeHints (
   DataHandler dh,
   long flags,
   long bandwidthPriority,
   TimeScale scale,
   TimeValue minTime,
   TimeValue maxTime
);
```

Parameters

dh

A data handler component.

flags

Undocumented

bandwidthPriority

Undocumented

scale

Undocumented

minTime

Undocumented

maxTime

Undocumented

Return Value

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

Version Notes

Introduced in QuickTime 5.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`QuickTimeComponents.h`

DataHTask

Cedes processor time to your data handler.

```
ComponentResult DataHTask (
   DataHandler dh
);
```

Parameters

dh

Identifies the calling program's connection to your data handler component.

Return Value

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

Discussion

This function is essentially analogous to the Movie Toolbox's `MoviesTask` function. Client programs call this function in order to give your data handler component time to do its work. Because client programs will call this function frequently, and especially so during movie playback or capture, your data handler should return control quickly to the client program.

Special Considerations

Applications should call this function often so that your handler has enough time to do its work.

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

qtdataref

qtdataref.win

Declared In

`QuickTimeComponents.h`

DataHUpdateMovie

Updates the movie for a data handler's current data reference.

```
ComponentResult DataHUpdateMovie (
   DataHandler dh,
   Movie theMovie,
   short id
);
```

Parameters

dh

A data handler component.

theMovie

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

id

Specifies the resource containing the movie data.

Return Value

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

Version Notes

Introduced in QuickTime 3 or earlier.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`QuickTimeComponents.h`

DataHUseTemporaryDataRef

Undocumented

```
ComponentResult DataHUseTemporaryDataRef (
   DataHandler dh,
   long inFlags
);
```

Parameters

dh

A data handler component.

inFlags

Undocumented

Return Value

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

Version Notes

Introduced in QuickTime 6.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`QuickTimeComponents.h`

DataHWrite

Writes data to its current data reference.

```
ComponentResult DataHWrite (
   DataHandler dh,
   Ptr data,
   long offset,
   long size,
   DataHCompletionUPP completion,
   long refCon
);
```

Parameters

dh

Identifies the calling program's connection to your data handler component.

data

Specifies a pointer to the data to be written. Client programs should lock the memory area holding this data, allowing your component's `DataHWrite` function to move memory.

offset

The offset (in bytes) to the location in the current data reference at which to write the data.

size

The number of bytes to write.

completion

A pointer to a data-handler completion function, described in `DataHCompletionProc`. The client program must provide a completion routine for all asynchronous write requests. For synchronous requests, client programs should set this parameter to `NIL`. When your data handler finishes with the client program's write request, your component must call this routine even if the request fails. Your component should pass the reference constant that the client program provided with the `refCon` parameter.

refCon

A reference constant that your data handler component should provide to the data-handler completion function specified with the `completion` parameter. For synchronous operations, client programs should set this parameter to 0.

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

Discussion
This function provides both a synchronous and an asynchronous write interface. Synchronous write operations work like the `DataHPutData` (page 56) function; the data handler component returns control to the client program only after it has serviced the write request. Asynchronous write operations allow client programs to queue write requests. Your data handler queues the request and immediately returns control to the calling program. After your component actually writes the data, it calls the client program's data-handler completion function.

Special Considerations

Note that some data handlers may not support write operations. For example, some shared devices, such as a CD-ROM "jukebox," may be read-only devices. As a result, it is very important that your data handler correctly report its write capabilities to client programs.

Version Notes
Introduced in QuickTime 3 or earlier.

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

Related Sample Code

ElectricImageComponent

ElectricImageComponent.win

qtdataref

ThreadsImporter

ThreadsImportMovie

Declared In

`QuickTimeComponents.h`

DataHWrite64

Provides a 64-bit version of DataHWrite.

```
ComponentResult DataHWrite64 (
   DataHandler dh,
   Ptr data,
   const wide *offset,
   long size,
   DataHCompletionUPP completion,
   long refCon
);
```

Parameters

dh

Identifies the calling program's connection to your data handler component.

data

Specifies a pointer to the data to be written. Client programs should lock the memory area holding this data, allowing your component's `DataHWrite` function to move memory.

offset

A pointer to the offset (in bytes) of the location in the current data reference at which to write the data.

size

The number of bytes to write.

completion

A pointer to a data-handler completion function, described in `DataHCompletionProc`. The client program must provide a completion routine for all asynchronous write requests. For synchronous requests, client programs should set this parameter to `NIL`. When your data handler finishes with the client program's write request, your component must call this routine even if the request fails. Your component should pass the reference constant that the client program provided with the `refCon` parameter.

refCon

A reference constant that your data handler component should provide to the data-handler completion function specified with the `completion` parameter. For synchronous operations, client programs should set this parameter to 0.

Return Value

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

Discussion

The only difference between this function and `DataHWrite` (page 71) is that the `offset` parameter is a 64-bit integer instead of a 32-bit integer.

Special Considerations

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

Version Notes

Introduced in QuickTime 4.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`QuickTimeComponents.h`

DisposeCDataHandlerUPP

Disposes of a CDataHandlerUPP pointer.

```
void DisposeCDataHandlerUPP (
   CDataHandlerUPP userUPP
);
```

Parameters

`userUPP`

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

Version Notes

Introduced in QuickTime 6.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`QuickTimeComponents.h`

DisposeCharDataHandlerUPP

Disposes of a CharDataHandlerUPP pointer.

```
void DisposeCharDataHandlerUPP (
   CharDataHandlerUPP userUPP
);
```

Parameters

`userUPP`

A `CharDataHandlerUPP` pointer.

Version Notes

Introduced in QuickTime 5.

Availability

Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

DisposeCommentHandlerUPP

Disposes of a CommentHandlerUPP pointer.

```
void DisposeCommentHandlerUPP (
   CommentHandlerUPP userUPP
);
```

Parameters

`userUPP`
A `CommentHandlerUPP` pointer.

Version Notes
Introduced in QuickTime 5.

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

Declared In
`QuickTimeComponents.h`

DisposeDataHCompletionUPP

Disposes of a DataHCompletionUPP pointer.

```
void DisposeDataHCompletionUPP (
   DataHCompletionUPP userUPP
);
```

Parameters

`userUPP`
A `DataHCompletionUPP` pointer. See `Universal Procedure Pointers`.

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

Version Notes
Introduced in QuickTime 4.1.

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

Related Sample Code
qtdataref
qtdataref.win
qtfiletransfer
ThreadsImporter
ThreadsImportMovie

Declared In
`QuickTimeComponents.h`

DisposeEndDocumentHandlerUPP

Disposes of an EndDocumentHandlerUPP pointer.

```
void DisposeEndDocumentHandlerUPP (
   EndDocumentHandlerUPP userUPP
);
```

Parameters

userUPP

An `EndDocumentHandlerUPP` pointer.

Version Notes
Introduced in QuickTime 5.

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

Declared In
`QuickTimeComponents.h`

DisposeEndElementHandlerUPP

Disposes of an EndElementHandlerUPP pointer.

```
void DisposeEndElementHandlerUPP (
   EndElementHandlerUPP userUPP
);
```

Parameters

userUPP

An `EndElementHandlerUPP` pointer.

Version Notes
Introduced in QuickTime 5.

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

Declared In
`QuickTimeComponents.h`

DisposePreprocessInstructionHandlerUPP

Disposes of a PreprocessInstructionHandlerUPP pointer.

```
void DisposePreprocessInstructionHandlerUPP (
   PreprocessInstructionHandlerUPP userUPP
);
```

Parameters

userUPP

A `PreprocessInstructionHandlerUPP` pointer.

Version Notes
Introduced in QuickTime 5.

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

Declared In
`QuickTimeComponents.h`

DisposeStartDocumentHandlerUPP

Disposes of a StartDocumentHandlerUPP pointer.

```
void DisposeStartDocumentHandlerUPP (
   StartDocumentHandlerUPP userUPP
);
```

Parameters

userUPP

A `StartDocumentHandlerUPP` pointer.

Version Notes
Introduced in QuickTime 5.

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

Declared In
`QuickTimeComponents.h`

DisposeStartElementHandlerUPP

Disposes of a StartElementHandlerUPP pointer.

```
void DisposeStartElementHandlerUPP (
   StartElementHandlerUPP userUPP
);
```

Parameters

userUPP

A `StartElementHandlerUPP` pointer.

Version Notes
Introduced in QuickTime 5.

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

Declared In
`QuickTimeComponents.h`

DisposeVdigIntUPP

Disposes of a VdigIntUPP pointer.

```
void DisposeVdigIntUPP (
   VdigIntUPP userUPP
);
```

Parameters

userUPP

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

Return Value

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

Version Notes

Introduced in QuickTime 4.1.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`QuickTimeComponents.h`

NewCDataHandlerUPP

Allocates a Universal Procedure Pointer for the CDataHandlerProc callback.

```
CDataHandlerUPP NewCDataHandlerUPP (
   CDataHandler userRoutine
);
```

Parameters

userRoutine

Undocumented

Return Value

A new UPP; see `Universal Procedure Pointers`.

Version Notes

Introduced in QuickTime 6.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`QuickTimeComponents.h`

NewCharDataHandlerUPP

Undocumented

```
CharDataHandlerUPP NewCharDataHandlerUPP (
   CharDataHandler userRoutine
);
```

Parameters

userRoutine

Undocumented

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

Version Notes
Introduced in QuickTime 5.

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

Declared In
`QuickTimeComponents.h`

NewCommentHandlerUPP

Undocumented

```
CommentHandlerUPP NewCommentHandlerUPP (
   CommentHandler userRoutine
);
```

Parameters

`userRoutine`

Undocumented

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

Version Notes
Introduced in QuickTime 5.

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

Declared In
`QuickTimeComponents.h`

NewDataHCompletionUPP

Allocates a Universal Procedure Pointer for the DataHCompletionProc callback.

```
DataHCompletionUPP NewDataHCompletionUPP (
   DataHCompletionProcPtr userRoutine
);
```

Parameters

`userRoutine`

A pointer to your application-defined function.

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

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

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

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

Related Sample Code
qtdataref

qtdataref.win

qtfiletransfer

ThreadsImporter

ThreadsImportMovie

Declared In
`QuickTimeComponents.h`

NewEndDocumentHandlerUPP

Undocumented

```
EndDocumentHandlerUPP NewEndDocumentHandlerUPP (
   EndDocumentHandler userRoutine
);
```

Parameters

`userRoutine`
  Undocumented

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

Version Notes
Introduced in QuickTime 5.

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

Declared In
`QuickTimeComponents.h`

NewEndElementHandlerUPP

Undocumented

```
EndElementHandlerUPP NewEndElementHandlerUPP (
   EndElementHandler userRoutine
);
```

Parameters

`userRoutine`
  Undocumented

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

Version Notes
Introduced in QuickTime 5.

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

Declared In
`QuickTimeComponents.h`

NewPreprocessInstructionHandlerUPP

Undocumented

```
PreprocessInstructionHandlerUPP NewPreprocessInstructionHandlerUPP (
   PreprocessInstructionHandler userRoutine
);
```

Parameters

`userRoutine`
Undocumented

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

Version Notes
Introduced in QuickTime 5.

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

Declared In
`QuickTimeComponents.h`

NewStartDocumentHandlerUPP

Undocumented

```
StartDocumentHandlerUPP NewStartDocumentHandlerUPP (
   StartDocumentHandler userRoutine
);
```

Parameters

`userRoutine`
Undocumented

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

Version Notes
Introduced in QuickTime 5.

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

Declared In
`QuickTimeComponents.h`

NewStartElementHandlerUPP

Undocumented

```
StartElementHandlerUPP NewStartElementHandlerUPP (
   StartElementHandler userRoutine
);
```

Parameters

`userRoutine`

Undocumented

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

Version Notes
Introduced in QuickTime 5.

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

Declared In
`QuickTimeComponents.h`

NewVdigIntUPP

Allocates a Universal Procedure Pointer for the VdigIntProc callback.

```
VdigIntUPP NewVdigIntUPP (
   VdigIntProcPtr userRoutine
);
```

Parameters

`userRoutine`

A pointer to your application-defined function.

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

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

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

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

Declared In
`QuickTimeComponents.h`

XMLParseAddAttribute

Undocumented

```
ComponentResult XMLParseAddAttribute (
   ComponentInstance aParser,
   UInt32 elementID,
   UInt32 nameSpaceID,
   char *attributeName,
   UInt32 *attributeID
);
```

Parameters

aParser

Undocumented

elementID

Undocumented

nameSpaceID

Undocumented

attributeName

Undocumented

attributeID

Undocumented

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

Version Notes
Introduced in QuickTime 5.

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

Declared In
`QuickTimeComponents.h`

XMLParseAddAttributeAndValue

Undocumented

```
ComponentResult XMLParseAddAttributeAndValue (
   ComponentInstance aParser,
   UInt32 elementID,
   UInt32 nameSpaceID,
   char *attributeName,
   UInt32 *attributeID,
   UInt32 attributeValueKind,
   void *attributeValueKindInfo
);
```

Parameters

aParser

Undocumented

`elementID`
: *Undocumented*

`nameSpaceID`
: *Undocumented*

`attributeName`
: *Undocumented*

`attributeID`
: *Undocumented*

`attributeValueKind`
: *Undocumented*

`attributeValueKindInfo`
: *Undocumented*

Return Value

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

Version Notes

Introduced in QuickTime 5.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`QuickTimeComponents.h`

XMLParseAddAttributeValueKind

Undocumented

```
ComponentResult XMLParseAddAttributeValueKind (
   ComponentInstance aParser,
   UInt32 elementID,
   UInt32 attributeID,
   UInt32 attributeValueKind,
   void *attributeValueKindInfo
);
```

Parameters

`aParser`
: *Undocumented*

`elementID`
: *Undocumented*

`attributeID`
: *Undocumented*

`attributeValueKind`
: *Undocumented*

`attributeValueKindInfo`
: *Undocumented*

Return Value

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

Version Notes
Introduced in QuickTime 5.

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

Declared In
`QuickTimeComponents.h`

XMLParseAddElement

Undocumented

```
ComponentResult XMLParseAddElement (
   ComponentInstance aParser,
   char *elementName,
   UInt32 nameSpaceID,
   UInt32 *elementID,
   long elementFlags
);
```

Parameters

aParser
: *Undocumented*

elementName
: *Undocumented*

nameSpaceID
: *Undocumented*

elementID
: *Undocumented*

elementFlags
: *Undocumented*

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

Version Notes
Introduced in QuickTime 5.

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

Declared In
`QuickTimeComponents.h`

XMLParseAddMultipleAttributes

Undocumented

```
ComponentResult XMLParseAddMultipleAttributes (
   ComponentInstance aParser,
   UInt32 elementID,
   UInt32 *nameSpaceIDs,
   char *attributeNames,
   UInt32 *attributeIDs
);
```

Parameters

aParser

Undocumented

elementID

Undocumented

nameSpaceIDs

Undocumented

attributeNames

Undocumented

attributeIDs

Undocumented

Return Value

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

Version Notes

Introduced in QuickTime 5.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`QuickTimeComponents.h`

XMLParseAddMultipleAttributesAndValues

Undocumented

```
ComponentResult XMLParseAddMultipleAttributesAndValues (
   ComponentInstance aParser,
   UInt32 elementID,
   UInt32 *nameSpaceIDs,
   char *attributeNames,
   UInt32 *attributeIDs,
   UInt32 *attributeValueKinds,
   void **attributeValueKindInfos
);
```

Parameters

aParser

Undocumented

elementID

Undocumented

nameSpaceIDs

    Undocumented

attributeNames

    Undocumented

attributeIDs

    Undocumented

attributeValueKinds

    Undocumented

attributeValueKindInfos

    Undocumented

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

Version Notes
Introduced in QuickTime 5.

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

Declared In
`QuickTimeComponents.h`

XMLParseAddNameSpace

Undocumented

```
ComponentResult XMLParseAddNameSpace (
   ComponentInstance aParser,
   char *nameSpaceURL,
   UInt32 *nameSpaceID
);
```

Parameters

aParser

    Undocumented

nameSpaceURL

    Undocumented

nameSpaceID

    Undocumented

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

Version Notes
Introduced in QuickTime 5.

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

Declared In
`QuickTimeComponents.h`

XMLParseDataRef

Undocumented

```
ComponentResult XMLParseDataRef (
   ComponentInstance aParser,
   Handle dataRef,
   OSType dataRefType,
   long parseFlags,
   XMLDoc *document
);
```

Parameters

aParser

  Undocumented

dataRef

  Undocumented

dataRefType

  Undocumented

parseFlags

  Undocumented

document

  Undocumented

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

Version Notes
Introduced in QuickTime 5.

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

Declared In
`QuickTimeComponents.h`

XMLParseDisposeXMLDoc

Undocumented

```
ComponentResult XMLParseDisposeXMLDoc (
   ComponentInstance aParser,
   XMLDoc document
);
```

Parameters

aParser

  Undocumented

document

  Undocumented

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

Version Notes
Introduced in QuickTime 5.

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

Declared In
`QuickTimeComponents.h`

XMLParseFile

Undocumented

```
ComponentResult XMLParseFile (
   ComponentInstance aParser,
   ConstFSSpecPtr fileSpec,
   long parseFlags,
   XMLDoc *document
);
```

Parameters

aParser
    Undocumented

fileSpec
    Undocumented

parseFlags
    Undocumented

document
    Undocumented

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

Version Notes
Introduced in QuickTime 5.

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

Declared In
`QuickTimeComponents.h`

XMLParseGetDetailedParseError

Undocumented

```
ComponentResult XMLParseGetDetailedParseError (
   ComponentInstance aParser,
   long *errorLine,
   StringPtr errDesc
);
```

Parameters

aParser
: *Undocumented*

errorLine
: *Undocumented*

errDesc
: *Undocumented*

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

Version Notes
Introduced in QuickTime 5.

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

Declared In
`QuickTimeComponents.h`

XMLParseSetCDataHandler

Undocumented

```
ComponentResult XMLParseSetCDataHandler (
   ComponentInstance aParser,
   CDataHandlerUPP cdata
);
```

Parameters

aParser
: *Undocumented*

cdata
: *Undocumented*

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

Version Notes
Introduced in QuickTime 6.

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

Declared In
`QuickTimeComponents.h`

XMLParseSetCharDataHandler

Undocumented

```
ComponentResult XMLParseSetCharDataHandler (
   ComponentInstance aParser,
   CharDataHandlerUPP charData
);
```

Parameters

aParser

Undocumented

charData

Undocumented

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

Version Notes
Introduced in QuickTime 5.

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

Declared In
`QuickTimeComponents.h`

XMLParseSetCommentHandler

Undocumented

```
ComponentResult XMLParseSetCommentHandler (
   ComponentInstance aParser,
   CommentHandlerUPP comment
);
```

Parameters

aParser

Undocumented

comment

Undocumented

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

Version Notes
Introduced in QuickTime 5.

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

Declared In
`QuickTimeComponents.h`

XMLParseSetEndDocumentHandler

Undocumented

```
ComponentResult XMLParseSetEndDocumentHandler (
   ComponentInstance aParser,
   EndDocumentHandlerUPP endDocument
);
```

Parameters

aParser

 Undocumented

endDocument

 Undocumented

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

Version Notes
Introduced in QuickTime 5.

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

Declared In
`QuickTimeComponents.h`

XMLParseSetEndElementHandler

Undocumented

```
ComponentResult XMLParseSetEndElementHandler (
   ComponentInstance aParser,
   EndElementHandlerUPP endElement
);
```

Parameters

aParser

 Undocumented

endElement

 Undocumented

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

Version Notes
Introduced in QuickTime 5.

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

Declared In
`QuickTimeComponents.h`

XMLParseSetEventParseRefCon

Undocumented

```
ComponentResult XMLParseSetEventParseRefCon (
   ComponentInstance aParser,
   long refcon
);
```

Parameters

aParser

Undocumented

refcon

Undocumented

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

Version Notes
Introduced in QuickTime 5.

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

Declared In
`QuickTimeComponents.h`

XMLParseSetOffsetAndLimit

Undocumented

```
ComponentResult XMLParseSetOffsetAndLimit (
   ComponentInstance aParser,
   UInt32 offset,
   UInt32 limit
);
```

Parameters

aParser

Undocumented

offset

Undocumented

limit

Undocumented

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

Version Notes
Introduced in QuickTime 5.

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

Declared In
`QuickTimeComponents.h`

XMLParseSetPreprocessInstructionHandler

Undocumented

```
ComponentResult XMLParseSetPreprocessInstructionHandler (
   ComponentInstance aParser,
   PreprocessInstructionHandlerUPP preprocessInstruction
);
```

Parameters

`aParser`
    Undocumented

`preprocessInstruction`
    Undocumented

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

Version Notes
Introduced in QuickTime 5.

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

Declared In
`QuickTimeComponents.h`

XMLParseSetStartDocumentHandler

Undocumented

```
ComponentResult XMLParseSetStartDocumentHandler (
   ComponentInstance aParser,
   StartDocumentHandlerUPP startDocument
);
```

Parameters

`aParser`
    Undocumented

`startDocument`
    Undocumented

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

Version Notes
Introduced in QuickTime 5.

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

Declared In
`QuickTimeComponents.h`

XMLParseSetStartElementHandler

Undocumented

```
ComponentResult XMLParseSetStartElementHandler (
   ComponentInstance aParser,
   StartElementHandlerUPP startElement
);
```

Parameters

`aParser`
: *Undocumented*

`startElement`
: *Undocumented*

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

Version Notes
Introduced in QuickTime 5.

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

Declared In
`QuickTimeComponents.h`

Callbacks

DataHCompletionProc

Called upon completion of a read or write operation.

```
typedef void (*DataHCompletionProcPtr) (Ptr request, long refcon, OSErr err);
```

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

```
void MyDataHCompletionProc (
    Ptr      request,
    long     refcon,
    OSErr    err );
```

Parameters

`request`
: Specifies a pointer to the data that was associated with the read request `DataHScheduleData` (page 60) or write request `DataHWrite` (page 71). The client program uses this pointer to determine which request has completed.

refcon

A reference constant that the client program supplied to your data handler component when it made the original request.

err

Indicates the success or failure of the operation. If the operation succeeded, set this parameter to 0. Otherwise, specify an appropriate error code.

Discussion

Data handler completion functions are guaranteed to be called at non-interrupt time. This means that you can safely call functions that are not interrupt-safe, such as `DataHScheduleData` (page 60), from within a completion function.

Declared In

`QuickTimeComponents.h`

Data Types

ConstFSSpecPtr

Represents a type used by the Data Components API.

```
typedef const FSSpec * ConstFSSpecPtr;
```

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Files.h`

DataCodecComponent

Represents a type used by the Data Components API.

```
typedef ComponentInstance DataCodecComponent;
```

Availability

Available in Mac OS X v10.0 and later.

Declared In

`QuickTimeComponents.h`

DataHCompletionUPP

Represents a type used by the Data Components API.

```
typedef STACK_UPP_TYPE(DataHCompletionProcPtr) DataHCompletionUPP;
```

Availability

Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.h`

DataHFileTypeOrderingHandle

Represents a type used by the Data Components API.

```
typedef DataHFileTypeOrderingPtr * DataHFileTypeOrderingHandle;
```

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

Declared In
`QuickTimeComponents.h`

DataHFileTypeOrderingPtr

Represents a type used by the Data Components API.

```
typedef OSType * DataHFileTypeOrderingPtr;
```

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

Declared In
`QuickTimeComponents.h`

DataHSchedulePtr

Represents a type used by the Data Components API.

```
typedef DataHScheduleRecord * DataHSchedulePtr;
```

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

Declared In
`QuickTimeComponents.h`

DataHScheduleRecord

Provides scheduling information for scheduled reads.

```
struct DataHScheduleRecord {
     TimeRecord    timeNeededBy;
     long          extendedID;
     long          extendedVers;
     Fixed         priority;
 };
```

Fields

`timeNeededBy`

Discussion

Specifies the time at which your data handler must deliver the requested data to the calling program. This time value is relative to the time base that is contained in this time record. During pre-roll operations, the Movie Toolbox may use special values in certain time record fields. The time record fields in question are the `scale` and value fields. By correctly interpreting the `values` of these fields, your data handler can queue up the pre-roll read requests in the most efficient way for its device.

`extendedID`

Discussion

A constant (see below) that indicates the type of data that follows in the remainder of the structure. See these constants:

    kDataHExtendedSchedule

`extendedVers`

Discussion

Reserved. This field should always be set to 0.

`priority`

Discussion

Indicates the relative importance of the data request. Client programs assign a value of 100.0 to data requests the must be delivered. Lower values indicate relatively less critical data. If your data handler must accommodate bandwidth limitations when delivering data, your component may use this value as an indication of which requests can be dropped with the least impact on the client program. As an example, consider using priorities in a frame-differenced movie. Key frames might have priority values of 100.0, indicating that they are essential to proper playback. As you move through the frames following a key frame, each successive frame might have a lower priority value. Once you drop a frame, you must drop all successive frames of equal or lower priority until you reach another key frame, because each of these frames would rely on the dropped one for some image data.

Discussion

There are two types of preroll read operations. The first type is a required read; that is, the Movie Toolbox requires that the read operation be satisfied before the movie starts playing. The second type is an optional read. If your data handler can satisfy the read operation as part of the pre-roll operation, it should do so. Otherwise, your data handler may satisfy the request at a specified time while the movie is playing. The Movie Toolbox indicates that a preroll read request is required by setting the `scale` field of the time record to -1. This literally means that the request is scheduled for a time that is infinitely far into the future. Your data handler should collect all such read requests, order them most efficiently for your device, and process them when the Movie Toolbox calls your component's `DataHFinishData` (page 28) function. For optional preroll read requests, the Movie Toolbox sets the `scale` field properly, but negates the `contents` of the `value` field. Your data handler has the option of delivering the data for this request with the required data, if that can be done efficiently. Otherwise, your data handler may deliver the data at its schedule time. You determine the scheduled time by negating the `contents` of the `value` field (that is, multiplying by -1).

Declared In

`QuickTimeComponents.h`

DataHVolumeList

Represents a type used by the Data Components API.

```
typedef DataHVolumeListPtr * DataHVolumeList;
```

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

Declared In
`QuickTimeComponents.h`

DataHVolumeListPtr

Represents a type used by the Data Components API.

```
typedef DataHVolumeListRecord * DataHVolumeListPtr;
```

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

Declared In
`QuickTimeComponents.h`

XMLDoc

Represents a type used by the Data Components API.

```
typedef XMLDocRecord * XMLDoc;
```

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

Declared In
`QuickTimeComponents.h`

XMLDocRecord

Undocumented

```
struct XMLDocRecord {
     void *        xmlDataStorage;
     XMLElement    rootElement;
 };
```

Fields

`xmlDataStorage`

Discussion
Undocumented

`rootElement`

Discussion
Undocumented

Declared In
`QuickTimeComponents.h`

Constants

kDataHCanRead

Constants grouped with kDataHCanRead.

```
enum {
  kDataHCanRead                  = 1L << 0,
  kDataHSpecialRead              = 1L << 1,
  kDataHSpecialReadFile          = 1L << 2,
  kDataHCanWrite                 = 1L << 3,
  kDataHSpecialWrite             = 1 << 4,
  kDataHSpecialWriteFile         = 1 << 5,
  kDataHCanStreamingWrite        = 1 << 6,
  kDataHMustCheckDataRef         = 1 << 7
};
```

Constants

`kDataHCanRead`

Indicates that your data handler can read from the volume.

Available in Mac OS X v10.0 and later.

Declared in `QuickTimeComponents.h`.

`kDataHSpecialRead`

Indicates that your data handler can read from the volume using a specialized method. For example, your data handler might support access to networked multimedia servers using a special protocol. In that case, your component would set this flag to 1 whenever the volume resides on a supported server.

Available in Mac OS X v10.0 and later.

Declared in `QuickTimeComponents.h`.

`kDataHSpecialReadFile`

Reserved for use by Apple.

Available in Mac OS X v10.0 and later.

Declared in `QuickTimeComponents.h`.

`kDataHCanWrite`

Indicates that your data handler can write data to the volume. In particular, use this flag to indicate that your data handler's `DataHPutData` (page 56) function will work with this volume.

Available in Mac OS X v10.0 and later.

Declared in `QuickTimeComponents.h`.

`kDataHSpecialWrite`

Indicates that your data handler can write to the volume using a specialized method. As with the `kDataHSpecialRead` flag, your data handler would use this flag to indicate that your component can access the volume using specialized support (for example, special network protocols).

Available in Mac OS X v10.0 and later.

Declared in `QuickTimeComponents.h`.

`kDataHCanStreamingWrite`

Indicates that your data handler can support the special write functions for capturing movie data when writing to this volume.

Available in Mac OS X v10.0 and later.

Declared in `QuickTimeComponents.h`.

Declared In
`QuickTimeComponents.h`

DataHScheduleRecord Values

Constants passed to DataHScheduleRecord.

```
enum {
  kDataHExtendedSchedule        = 'xtnd'
};
```

Declared In
`QuickTimeComponents.h`

DataHGetFileTypeOrdering Values

Constants passed to DataHGetFileTypeOrdering.

```
enum {
  kDataHFileTypeMacOSFileType   = 'ftyp',
  kDataHFileTypeExtension       = 'fext',
  kDataHFileTypeMIME            = 'mime'
};
```

Declared In
`QuickTimeComponents.h`

DataHGetInfoFlags Values

Constants passed to DataHGetInfoFlags.

```
enum {
  kDataHInfoFlagNeverStreams    = 1 << 0, /* set if this data handler doesn't
stream*/
  kDataHInfoFlagCanUpdateDataRefs = 1 << 1, /* set if this data handler might update
 data reference*/
  kDataHInfoFlagNeedsNetworkBandwidth = 1 << 2 /* set if this data handler may need
 to occupy the network*/
};
```

Declared In
`QuickTimeComponents.h`

DataHSetMovieUsageFlags Values

Constants passed to DataHSetMovieUsageFlags.

```
enum {
  kDataHMovieUsageDoAppendMDAT  = 1L << 0 /* if set, datahandler should append wide
 and mdat atoms in append call*/
};
```

Declared In

`QuickTimeComponents.h`

Document Revision History

This table describes the changes to *Data Components Reference for QuickTime*.

Date	Notes
2006-05-23	New document, based on previously published material, that describes the API for QuickTime data components.

Index