QuickTime Atoms and Resources Reference

QuickTime

2006-05-23

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

Simultaneously published in the United States and Canada.

Contents

QuickTime Atoms and Resources Reference

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

Overview

This reference covers the API details of QuickTime atoms and public resource types.

QuickTime Atoms

Atoms

0x00000000 Terminates an audio atom list.

```
struct AudioTerminatorAtom {
    long      size;
    OSType    atomType;
};
```

Fields
`size`

Discussion
The size in bytes of this atom structure.

`atomType`

Discussion
Constant `kAudioTerminatorAtomType`.

Programming Info
C interface file: `Sound.h`

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

0x00000001 A sprite property matrix atom.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields
`atomType`

Discussion
0x00000001.

`data`

Discussion
The sprite matrix property, a structure of type `MatrixRecord`.

Parent Atom

`'sprt'` (page 66)
Parent atom can contain only one atom of this type.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

0x00000004 A sprite visible property atom.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields
`atomType`

Discussion
0x00000004.

`data`

Discussion
The sprite visible property, of type `short`.

Parent Atom

`'sprt'` (page 66)
Parent atom can contain only one atom of this type.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

0x00000005 A sprite property layer atom.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields
`atomType`

Discussion
0x00000005.

`data`

Discussion
The sprite layer property, of type `short`.

Parent Atom

`'sprt'` (page 66)
Parent atom can contain only one atom of this type.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

0x00000006 A sprite graphics mode property atom.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields
`atomType`

Discussion
0x00000006.

`data`

Discussion
The sprite graphics mode property.

Parent Atom

`'sprt'` (page 66)

Parent atom can contain only one atom of this type.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

0x00000064 A sprite image index atom.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields

`atomType`

Discussion

0x00000064.

`data`

Discussion

The sprite image index property, of type `short`.

Parent Atom

`'sprt'` (page 66)

Parent atom can contain only one atom of this type.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

0x00000065 A sprite background color property atom.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields

`atomType`

Discussion

0x00000065.

`data`

Discussion

The sprite background color property, a structure of type `RGBColor`.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

0x00000066 A sprite property offscreen bit depth atom.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields

`atomType`

Discussion

0x00000066.

`data`

Discussion
The sprite offscreen bit depth property, of type `short`.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

0x00000067 A sprite property sample format atom.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields
`atomType`

Discussion
0x00000067.

`data`

Discussion
The sprite sample format property, of type `short`.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'AllF' User data list entry atom to play all frames.

```
struct MoviesUserData {
    long    size;
    long    udType;
    char    data[1];
};
```

Fields
`size`

Discussion
The size in bytes of this atom structure.

`udType`

Discussion
`'AllF'`.

`data`

Discussion
A byte indicating that all frames of video should be played, regardless of timing.

Parent Atom

`'udta'` (page 84)
Parent atom can contain only one atom of this type.

Programming Info
C interface file: `MoviesFormat.h`

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'beha' Defines sprite behavior.

This is a QT container atom; it is not declared in the header files. You can create it with `QTNewAtomContainer` and insert it with `QTInsertChild`, using the following parameter:

Fields
`atomType`

Discussion
Value is `'beha'`.

Optional Child Atoms

`'imag'` (page 38)
A sprite image atom.
`'crsr'` (page 20)
Color custom cursor child atom.
`'sstr'` (page 67)
Specifies the ID of a string variable, contained in a sprite track, to display in the status area of the browser.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'chap' Chapter or scene list track reference type atom.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields
`atomType`

Discussion
Constant `kTrackReferenceChapterList`, designating atom type `'chap'`.

`data`

Discussion
A list of track ID values (32-bit integers) specifying the related tracks. Note that a track ID value can be set to 0 to indicate an unused entry in the atom. Doing this can be more convenient than deleting the reference.

Parent Atom

`'tref'` (page 80)
Parent atom can contain only one atom of this type.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'clip' Defines a clipping region.

```
struct ClippingAtom {
    long       size;
    long       atomType;
    RgnAtom    aRgnClip;
};
```

Fields
`size`

Discussion
The size in bytes of this atom structure.

`atomType`

Discussion
Constant `ClipAID`, designating atom type `'clip'`.

`aRgnClip`

Discussion
A `'crgn'` (page 20) atom that defines the clipping region.

Discussion
You can treat this atom either as a declared structure or as a QT atom, which you can create it with `QTInsertChild`.

Programming Info
C interface file: `MoviesFormat.h`

See Also

For the atoms that may contain this atom, see `'moov'` (page 52) and `'trak'` (page 79). For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'clon' Contains information about a track clone.

```
struct CloneAtom {
    long           size;
    long           atomType;
    CloneRecord    cloneInfo;
};
```

Fields
`size`

Discussion
The size in bytes of this atom structure.

`atomType`

Discussion
Value is `'clon'`.

`cloneInfo`

Discussion
A `CloneRecord` structure.

See Also

See the `CloneRecord` structure and `AddClonedTrackToMovie`. For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'cmov' Contains a compressed movie.

This is a QT atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameter:

Fields
`atomType`

Discussion
Constant `CompressedMovieAID`, designating atom type `'cmov'`.

Parent Atom

`'moov'` (page 52)
Parent atom can contain only one atom of this type.

Required Child Atoms

`'dcom'` (page 23)
Indicates the compression algorithm used to compress a movie. Only one allowed.
`'cmvd'` (page 13)
Stores the data for a compressed movie. Only one allowed.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'cmvd' Stores the data for a compressed movie.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields
`atomType`

Discussion
Constant `CompressedMovieDataAID`, designating atom type `'cmvd'`.

`data`

Discussion
An integer of type UInt32 that gives the length of the uncompressed movie in bytes, followed by the compressed movie data.

Parent Atom

`'cmov'` (page 12)
Parent atom can contain only one atom of this type.
'co64' A 64-bit version of the `'stco'` (page 69) atom.

For details, see `'stco'` (page 69).

Fields
`atomType`

Discussion
Constant STChunkOffset64AID, designating atom type 'co64'.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'©cpy' User data list entry atom: copyright information.

```
struct MoviesUserData {
    long    size;
    long    udType;
    char    data[1];
};
```

Fields
`size`

Discussion
The size in bytes of this atom structure.

`udType`

Discussion
Constant `kUserDataTextCopyright`, designating atom type `'©cpy'`.

`data`

Discussion
A string containing copyright information.

Parent Atom

`'udta'` (page 84)
Parent atom can contain only one atom of this type.

Programming Info

C interface file: `MoviesFormat.h`

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'©day' User data list entry atom: creation date.

```
struct MoviesUserData {
    long    size;
    long    udType;
    char    data[1];
};
```

Fields

`size`

Discussion
The size in bytes of this atom structure.

`udType`

Discussion
Constant `kUserDataTextCreationDate`, designating atom type `'©day'`.

`data`

Discussion
A string containing the creation date.

Parent Atom

`'udta'` (page 84)
Parent atom can contain only one atom of this type.

Programming Info

C interface file: `MoviesFormat.h`

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'©dir' User data list entry atom: name of movie's director.

```
struct MoviesUserData {
    long    size;
    long    udType;
    char    data[1];
};
```

Fields
`size`

Discussion
The size in bytes of this atom structure.

`udType`

Discussion
Constant `kUserDataTextDirector`, designating atom type `'©dir'`.

`data`

Discussion
A string containing the name of the movie's director.

Parent Atom

`'udta'` (page 84)
Parent atom can contain only one atom of this type.

Programming Info
C interface file: `MoviesFormat.h`

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'©ed1' User data list entry atom: edit date 1.

```
struct MoviesUserData {
    long    size;
    long    udType;
    char    data[1];
};
```

Fields
`size`

Discussion
The size in bytes of this atom structure.

`udType`

Discussion
Constant kUserDataTextEditDate1, designating atom type '©ed1'.

`data`

Discussion
A string containing the first edit date.

Parent Atom

`'udta'` (page 84)
Parent atom can contain only one atom of this type.

Discussion
Similar atoms of types '©ed2' through '©ed9' may contain other edit date strings.

Programming Info
C interface file: `MoviesFormat.h`

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'©fmt' User data list entry atom: indication of movie's format.

```
struct MoviesUserData {
    long    size;
    long    udType;
    char    data[1];
};
```

Fields
`size`
Discussion
The size in bytes of this atom structure.

`udType`
Discussion
Constant `kUserDataTextOriginalFormat`, designating atom type `'©fmt'`.

`data`
Discussion
A string indicating the movie's format.

Parent Atom

`'udta'` (page 84)
Parent atom can contain only one atom of this type.

Programming Info
C interface file: `MoviesFormat.h`

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'©inf' User data list entry atom: information about the movie.

```
struct MoviesUserData {
    long    size;
    long    udType;
    char    data[1];
};
```

Fields
`size`
Discussion
The size in bytes of this atom structure.

`udType`
Discussion
Constant `kUserDataTextInformation`, designating atom type `'©inf'`.

`data`

Discussion
A string containing information about the movie.

Parent Atom

`'udta'` (page 84)
Parent atom can contain only one atom of this type.

Programming Info
C interface file: `MoviesFormat.h`

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'©prd' User data list entry atom: name of movie's producer.

```
struct MoviesUserData {
    long    size;
    long    udType;
    char    data[1];
};
```

Fields
`size`

Discussion
The size in bytes of this atom structure.

`udType`

Discussion
Constant `kUserDataTextProducer`, designating atom type `'©prd'`.

`data`

Discussion
A string containing the name of the movie's producer.

Parent Atom

`'udta'` (page 84)
Parent atom can contain only one atom of this type.

Programming Info
C interface file: `MoviesFormat.h`

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'©prf' User data list entry atom: names of performers.

```
struct MoviesUserData {
    long    size;
    long    udType;
    char    data[1];
};
```

Fields
`size`

Discussion
The size in bytes of this atom structure.

`udType`

Discussion
Constant `kUserDataTextPerformers`, designating atom type `'©prf'`.

`data`

Discussion
A string containing names of the performers.

Parent Atom

`'udta'` (page 84)
Parent atom can contain only one atom of this type.

Programming Info

C interface file: `MoviesFormat.h`

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'©req' User data list entry atom: special hardware or software requirements.

```
struct MoviesUserData {
    long    size;
    long    udType;
    char    data[1];
};
```

Fields

`size`

Discussion
The size in bytes of this atom structure.

`udType`

Discussion
Constant `kUserDataTextSpecialPlaybackRequirements`, designating atom type `'©req'`.

`data`

Discussion
A string detailing special hardware or software requirements.

Parent Atom

`'udta'` (page 84)
Parent atom can contain only one atom of this type.

Programming Info

C interface file: `MoviesFormat.h`

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'©src' User data list entry atom: credits for those who provided movie source content.

```
struct MoviesUserData {
    long    size;
    long    udType;
    char    data[1];
};
```

Fields
`size`
Discussion
The size in bytes of this atom structure.

`udType`
Discussion
Constant `kUserDataTextOriginalSource`, designating atom type `'©src'`.

`data`
Discussion
A string containing credits for those who provided movie source content.

Parent Atom
`'udta'` (page 84)
Parent atom can contain only one atom of this type.

Programming Info
C interface file: `MoviesFormat.h`

See Also
For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'©wrt' User data list entry atom: name of movie's writer.

```
struct MoviesUserData {
    long    size;
    long    udType;
    char    data[1];
};
```

Fields
`size`
Discussion
The size in bytes of this atom structure.

`udType`
Discussion
Constant `kUserDataTextWriter`, designating atom type `'©wrt'`.

`data`
Discussion
A string containing the name of the movie's writer.

Parent Atom
`'udta'` (page 84)
Parent atom can contain only one atom of this type.

Programming Info
C interface file: `MoviesFormat.h`

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'crgn' Defines a clipping region.

```
struct RgnAtom {
    long     size;
    long     atomType;
    short    rgnSize;
    Rect     rgnBBox;
    char     data[1];
};
```

Fields

`size`

Discussion
The size in bytes of this atom structure.

`atomType`

Discussion
Constant `RgnClipAID`, designating atom type `'crgn'`.

`rgnSize`

Discussion
The size in bytes of the region.

`rgnBBox`

Discussion
The bounding box for the region.

`data`

Discussion
Additional data if the clipping region is not rectangular.

Parent Atom

`'clip'` (page 11)
Parent atom can contain only one atom of this type.

Programming Info

C interface file: `MoviesFormat.h`

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'crsr' Color custom cursor child atom.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields

`atomType`

Discussion
Constant `kSpriteCursorBehaviorAtomType`, designating atom type `'crsr'`.

`data`

Discussion
A cursor description.

Parent Atom

`'vrcp'` (page 88)
Parent atom can contain multiple atoms of this type.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'cspd' Contains the connection speed currently set in the QuickTime preferences.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields
`atomType`

Discussion
Constant `ConnectionSpeedPrefsType`, designating atom type `'cspd'`.

`data`

Discussion
The connection speed.

See Also

See the `GetQuickTimePreference` and `SetQuickTimePreference` functions. For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'ctab' Color table atom.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields
`atomType`

Discussion
Constant `ColorTableAID`, designating atom type `'ctab'`.

`data`

Discussion
A color table.

Parent Atom

`'moov'` (page 52)
Parent atom can contain only one atom of this type.

Discussion
Color table atoms define a list of preferred colors for displaying the movie on devices that support only 256 colors. The list may contain up to 256 colors.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'cufa' Non-standard cubic QTVR panorama data atom.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields
`atomType`
Discussion
Value is `'cufa'`.

`data`
Discussion
A `QTVRCubicFaceData` structure.

Discussion
Each entry in the `QTVRCubicFaceData` structure describes one face of the polyhedron being described.

See Also
For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'CURS' Custom cursor child atom.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields
`atomType`
Discussion
Constant `kQTVRCursorAtomType`, designating atom type `'CURS'`.

`data`
Discussion
A cursor description.

Parent Atom
`'vrcp'` (page 88)
Parent atom can contain multiple atoms of this type.
See Also
For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'cuvw' Cubic view atom.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields
`atomType`
Discussion
Value is `'cuvw'`.

`data`
Discussion
A `QTVRCubicViewAtom` structure.

See Also
For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'dasz' Data size atom.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields
`atomType`

Discussion
Constant `kQTTargetDataSize`, designating atom type `'dasz'`.

`data`

Discussion
A `QTTargetDataSize` structure.

Parent Atom

`'vide'` (page 86)
Parent atom can contain only one atom of this type.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'dcom' Indicates the compression algorithm used to compress a movie.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields
`atomType`

Discussion
Constant `DataCompressionAtomAID`, designating atom type `'dcom'`.

`data`

Discussion
A 32-bit constant (see below) that indicates which lossless algorithm was used to compress the movie contained in the parent atom.

Data Constants

`AppleDataCompressorSubType`
The Apple data compressor; value is `'adec'`.
`zlibDataCompressorSubType`
The zlib data compressor; value is `'zlib'`.

Parent Atom

`'cmov'` (page 12)
Parent atom can contain only one atom of this type.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'defi' A sprite image data reference atom.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields
`atomType`

Discussion
Constant `kSpriteImageDefaultImageIndexAtomType`, designating atom type `'defi'`.

`data`

Discussion
The image index of a traditional image, of type `short`, to use while waiting for the referenced image to load.

Parent Atom

`'imag'` (page 38)
Parent atom can contain only one atom of this type.

Discussion
You use the this atom type to specify that an image is referenced and how to access it. Its ID should be 1.

Version Notes
Added to QuickTime 4.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'desc' Graphics export description atom.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields
`atomType`

Discussion
Constant `kCustomHandlerDesc`, designating atom type `'desc'`.

`data`

Discussion
A nonterminated string containing a human-readable format name.

Parent Atom

`'expo'` (page 30)
Parent atom can contain only one atom of this type.

See Also

See the `GraphicsExportGetMIMETypeList` and `GraphicsImportGetExportImageTypeList` functions.
For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'dflt' Key frame shared-data atom.

This is a QT atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameter:

Fields
`atomType`

Discussion
Constant `kSpriteSharedDataAtomType`, designating atom type `'dflt'`.

Required Child Atoms

`'imct'` (page 39)
Only one allowed.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'dimm' Number of bytes of immediate data to be sent.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields

`atomType`

Discussion
Value is `'dimm'`.

`data`

Discussion
8-byte value.

Parent Atom

`'hinf'` (page 34)
Parent atom can contain only one atom of this type.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'dinf' Specifies where media data is stored.

```
struct DataInfoAtom {
    long           size;
    long           atomType;
    DataRefAtom    dataRef;
};
```

Fields

`size`

Discussion
The size in bytes of this atom structure.

`atomType`

Discussion
Constant `DataInfoAID`, designating atom type `'dinf'`.

`dataRef`

Discussion
A value that contains the data for this atom. The 4-byte `DataRefAtom` data type is private and is not documented.

Optional Child Atoms

`'dref'` (page 26)
Only one allowed.

Programming Info

C interface file: `MoviesFormat.h`

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'dmax' The largest packet duration.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields
`atomType`

Discussion
Value is `'dmax'`.

`data`

Discussion
4 bytes packet duration, in milliseconds.

Parent Atom

`'hinf'` (page 34)
Parent atom can contain only one atom of this type.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'dmed' Number of bytes from the media track to be sent.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields
`atomType`

Discussion
Value is `'dmed'`.

`data`

Discussion
8-byte value.

Parent Atom

`'hinf'` (page 34)
Parent atom can contain only one atom of this type.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'dref' Data reference atom.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields
`atomType`

Discussion
Constant `DataRefAID`, designating atom type `'dref'`.

`data`

Discussion
Data references.

Parent Atom

`'dinf'` (page 25)
Parent atom can contain only one atom of this type.

See Also

See the `AliasRecord` structure. For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'drep' Number of bytes of repeated data to be sent.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields
`atomType`

Discussion
Value is `'drep'`.

`data`

Discussion
8-byte value.

Parent Atom

`'hinf'` (page 34)
Parent atom can contain only one atom of this type.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'edts' Contains an atom that defines an edit list.

```
struct EditsAtom {
    long            size;
    long            atomType;
    EditListAtom    editList;
};
```

Fields
`size`

Discussion
The size in bytes of this atom structure.

`atomType`

Discussion
Constant `EditsAID`, designating atom type `'edts'`.

`editList`

Discussion
An `'elst'` (page 28) atom.

Parent Atom

`'trak'` (page 79)

Parent atom can contain only one atom of this type.

Required Child Atoms

`'elst'` (page 28)

Only one allowed.

Programming Info

C interface file: `MoviesFormat.h`

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'elst' Contains a list of edit segment definitions for a media.

```
struct EditListAtom {
    long            size;
    long            atomType;
    long            flags;
    long            numEntries;
    EditListType    editListTable[1];
};
```

Fields

`size`

Discussion

The size in bytes of this atom structure.

`atomType`

Discussion

Constant `EditListAID`, designating atom type `'elst'`.

`flags`

Discussion

One byte of version information followed by three bytes of flags. The flag bytes are not currently used.

`numEntries`

Discussion

The number of entries in `editListTable`.

`editListTable`

Discussion

An array of `EditListType` data structures, each of which locates and defines an edit segment within a media.

Parent Atom

`'edts'` (page 27)

Parent atom can contain only one atom of this type.

Discussion

You can use the edit list atom to tell QuickTime how to map from a time in a movie to a time in a media, and ultimately to each segment of the media's data.

Programming Info

C interface file: `MoviesFormat.h`

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'end ' Defines the ending offset of hypertext in a text stream.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields
`atomType`

Discussion
Value is `'end '` (the fourth character is a space).

`data`

Discussion
The ending offset of hypertext in a text stream.

Parent Atom

`'htxt'` (page 37)
Parent atom can contain only one atom of this type.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'enda' Determines the endian status of the sound component that interprets data contained in an audio atom list.

```
struct AudioEndianAtom {
    long      size;
    OSType    atomType;
    short     littleEndian;
};
```

Fields
`size`

Discussion
The size in bytes of this atom structure.

`atomType`

Discussion
Constant `kAudioEndianAtomType`, designating atom type `'enda'`.

`littleEndian`

Discussion
Set this field to TRUE if the audio component is to operate on little-endian data, and FALSE otherwise.

Programming Info

C interface file: `Sound.h`

See Also

To choose the sound component for an audio atom list, see the `'frma'` (page 31) atom. For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'expo' Defines a graphics export group.

This is a QT container atom; it is not declared in the header files. You can create it with `QTNewAtomContainer` and insert it with `QTInsertChild`, using the following parameters:

Fields
`atomType`

Discussion
Constant `kGraphicsExportGroup`, designating atom type `'expo'`.

Required Child Atoms

`'ftyp'` (page 32)
An `OSType` representing the exported file type.
`'ext '` (page 30)
A nonterminated string containing the suggested file extension for this format.
`'desc'` (page 24)
A nonterminated string containing a human-readable name for this format.

Optional Child Atoms

`'mime'[atom]` (page 48)
A nonterminated string containing the MIME type for this format.

See Also

See the `GraphicsImportGetExportImageTypeList` function. For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'ext ' Defines a graphics export extension.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields
`atomType`

Discussion
Constant `kGraphicsExportExtension`, designating atom type `'ext '` (the fourth character is a space).

`data`

Discussion
A nonterminated string containing a file extension.

Parent Atom

`'expo'` (page 30)
Parent atom can contain only one atom of this type.

See Also

See the `GraphicsImportGetExportImageTypeList` function. For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'flap' Extension to the `SoundDescription` structure.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields
`atomType`

Discussion
Constant `siSlopeAndIntercept`, designating atom type `'flap'`; see `Sound Information Selectors`.

`data`

Discussion
A `SoundSlopeAndInterceptRecord` structure.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'flov' Contains a floating-point variable for a sprite.

This is a QT container atom; it is not declared in the header files. You can create it with `QTNewAtomContainer` and insert it with `QTInsertChild`, using the following parameter:

Fields
`atomType`

Discussion
Value is `'flov'`.

Parent Atom

`'vars'` (page 86)
Contains variables for a sprite.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'free' Provides unused space in a movie file.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields
`atomType`

Discussion
Constant `FreeAtomType`, designating atom type `'free'`.

`data`

Discussion
Any number of bytes of free space.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'frma' Specifies which sound component is responsible for the atoms contained in an audio atom list.

```
struct AudioFormatAtom {
    long      size;
    OSType    atomType;
    OSType    format;
};
```

Fields
`size`

Discussion
The size in bytes of this atom structure.

`atomType`

Discussion
Constant `kAudioFormatAtomType`, designating atom type `'frma'`.

`format`

Discussion
A constant that identifies a sound component. See `Codec Identifiers`.

Programming Info
C interface file: `Sound.h`

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'ftyp' Defines a graphics export file type.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields
`atomType`

Discussion
Constant `kGraphicsExportFileType`, designating atom type `'ftyp'`.

`data`

Discussion
An `OSType` representing the exported file type.

Parent Atom

`'expo'` (page 30)
Parent atom can contain only one atom of this type.

See Also

See the `GraphicsImportGetExportImageTypeList` function. For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'gmhd' Contains a generic media information atom.

This is a QT container atom; it is not declared in the header files. You can create it with `QTNewAtomContainer` and insert it with `QTInsertChild`, using the following parameter:

Fields
`atomType`

Discussion
Constant `GenericMediaInfoHeaderAID`, designating atom type `'gmhd'`.

Parent Atom

`'minf'[generic]` (page 49)
Parent atom can contain only one atom of this type.

Required Child Atoms

`'gmin'` (page 33)
Provides data that is specific to a handler for media other than video or sound. Only one allowed.

Discussion
This atom is currently used only as a container for a `'gmin'` (page 33) atom.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'gmin' Provides data that is specific to a handler for media other than video or sound.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields

`atomType`

Discussion
Constant `GenericMediaInfoAID`, designating atom type `'gmin'`.

`data`

Discussion
Data required by the media handler that is designated by the `'hdlr'` (page 33) atom contained in the `'minf'[generic]` (page 49) atom that also contains the parent of this atom.

Parent Atom

`'gmhd'` (page 32)
Parent atom can contain any number of atoms of this type.

Discussion
This atom contains handler-specific information to support your use of a `'minf'[generic]` (page 49) atom. Note that the data in this atom is not used by RTP servers.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'hdlr' Specifies the component that is to interpret a media's data.

```
struct HandlerAtom {
    long                size;
    long                atomType;
    PublicHandlerInfo   hInfo;
};
```

Fields

`size`

Discussion
The size in bytes of this atom structure.

`atomType`

Discussion
Constant `HandlerAID`, designating atom type `'hdlr'`.

`hInfo`

Discussion
A `PublicHandlerInfo` structure, which contains the actual data for this atom.

Discussion
RTP servers ignore this atom's data when it is contained in a `'minf'[generic]` (page 49) atom.

Programming Info

C interface file: `MoviesFormat.h`

See Also

For the atoms that may contain this atom, see `'mdia'` (page 47), `'minf'[generic]` (page 49), `'minf'[sound]` (page 50), and `'minf'[video]` (page 51). For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'hinf' Contains statistics for the hint track.

This is a QT container atom; it is not declared in the header files. You can create it with `QTNewAtomContainer` and insert it with `QTInsertChild`, using the following parameter:

Fields
`atomType`

Discussion
Value is `'hinf'`.

Required Child Atoms

`'trpy'` (page 81)
Total number of bytes that will be sent, including 12-byte RTP headers but not including network headers. Only one allowed.
`'nump'` (page 55)
Total number of network packets that will be sent. Only one allowed.
`'tpyl'` (page 78)
Total number of bytes that will be sent, not including 12-byte RTP headers. Only one allowed.
`'maxr'` (page 46)
Maximum data rate. Only one allowed.
`'dmed'` (page 26)
Number of bytes from the media track to be sent. Only one allowed.
`'dimm'` (page 25)
Number of bytes of immediate data to be sent. Only one allowed.
`'drep'` (page 27)
Number of bytes of repeated data to be sent. Only one allowed.
`'tmin'` (page 78)
Smallest relative transmission time, in milliseconds. Only one allowed.
`'tmax'` (page 77)
Largest relative transmission time, in milliseconds. Only one allowed.
`'pmax'` (page 57)
Largest packet, in bytes, including 12-byte RTP header. Only one allowed.
`'dmax'` (page 26)
The largest packet duration. Only one allowed.
`'payt'` (page 56)
Payload type. Only one allowed.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'hint' Hint track reference type atom.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields
`atomType`

Discussion
Value is `'hint'`.

`data`

Discussion
A list of track ID values (32-bit integers) specifying the related tracks. Note that a track ID value can be set to 0 to indicate an unused entry in the atom. Doing this can be more convenient than deleting the reference.

Parent Atom
`'tref'` (page 80)
Parent atom can contain only one atom of this type.

See Also
For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'hlit' Defines the highlighted portion in text.

```
struct HiliteAtom {
    long    size;
    long    atomType;
    long    selStart;
    long    selEnd;
};
```

Fields
`size`

Discussion
The size in bytes of this atom structure.

`atomType`

Discussion
Value is `'hlit'`.

`selStart`

Discussion
Character number of highlighted selection start character.

`selEnd`

Discussion
Character number of highlighted selection end character.

Discussion
For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

Programming Info
C interface file: `MoviesFormat.h`

'hnti' Hint track user data atom.

This is a QT container atom; it is not declared in the header files. You can create it with `QTNewAtomContainer` and insert it with `QTInsertChild`, using the following parameter:

Fields
`atomType`

Discussion
Value is `'hnti'`.

Required Child Atoms

`'sdp '` (page 64)
SDP text for a hint track. Only one allowed.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'hots' A QTVR hot spot.

This is a QT container atom; it is not declared in the header files. You can create it with `QTNewAtomContainer` and insert it with `QTInsertChild`, using the following parameter:

Fields

`atomType`

Discussion
Constant `kQTVRHotSpotAtomType`, designating atom type `'hots'`.

Parent Atom

`'hspa'` (page 36)
Parent atom can contain any number of atoms of this type.

Required Child Atoms

`'hsin'` (page 36)
Contains general hot spot information. Only one allowed.
`'link'` (page 44)
Specific information about a link hot spot. Only one allowed.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'hsin' Contains general hot spot information.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields

`atomType`

Discussion
Constant `kQTVRHotSpotInfoAtomType`, designating atom type `'hsin'`.

`data`

Discussion
Hot spot information.

Parent Atom

`'hots'` (page 36)
Parent atom can contain only one atom of this type.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'hspa' Hot spot parent atom.

This is a QT container atom; it is not declared in the header files. You can create it with `QTNewAtomContainer` and insert it with `QTInsertChild`, using the following parameter:

Fields
`atomType`

Discussion
Constant `kQTVRHotSpotParentAtomType`, designating atom type `'hspa'`.

Parent Atom

`'vrnp'` (page 88)
Parent atom can contain only one atom of this type.

Required Child Atoms

`'hots'` (page 36)
A QTVR hot spot. Any number allowed.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'htxt' Hypertext in a text stream.

This is a QT container atom; it is not declared in the header files. You can create it with `QTNewAtomContainer` and insert it with `QTInsertChild`, using the following parameter:

Fields
`atomType`

Discussion
Value is `'htxt'`.

Parent Atom

`'wtxt'` (page 91)
Parent atom can contain multiple atoms of this type.

Required Child Atoms

`'strt'` (page 70)
Defines the starting offset of hypertext in a text stream. Only one allowed.
`'end '` (page 29)
Defines the ending offset of hypertext in a text stream. Only one allowed.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'idat' Image data atom.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields
`atomType`

Discussion
Constant `quickTimeImageFileImageDataAtom`, designating atom type `'idat'`.

`data`

Discussion
The image data.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'idsc' Image description atom.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields
`atomType`

Discussion
Constant `quickTimeImageFileImageDescriptionAtom`, designating atom type `'idsc'`.

`data`

Discussion
The image description.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'iicc' ColorSync profile atom.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields
`atomType`

Discussion
Constant `quickTimeImageFileColorSyncProfileAtom`, designating atom type `'iicc'`.

`data`

Discussion
A ColorSync profile.

Version Notes
This is a new optional atom in QuickTime 4.

See Also

See the `GraphicsExportSetColorSyncProfile` function. For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'imag' A sprite image atom.

This is a QT container atom; it is not declared in the header files. You can create it with `QTNewAtomContainer` and insert it with `QTInsertChild`, using the following parameter:

Fields
`atomType`

Discussion
Constant `kSpriteImageAtomType`, designating atom type `'imag'`.

Parent Atom

`'imct'` (page 39)
Parent atom can contain any number of atoms of this type.

Required Child Atoms

`'imda'` (page 39)
Contains sprite image data. Only one allowed.

Optional Child Atoms

`'name'[sprite]` (page 54)
A sprite name atom. Only one allowed.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'imap' An input map.

This is a QT container atom; it is not declared in the header files. You can create it with `QTNewAtomContainer` and insert it with `QTInsertChild`, using the following parameter:

Fields

`atomType`

Discussion

Constant `InputMapAID`, designating atom type `'imap'`.

Parent Atom

`'trak'` (page 79)
Parent atom can contain only one atom of this type.

Required Child Atoms

`' in'` (page 43)
`Track` input atom. Only one allowed.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'imct' A sprite image container atom.

This is a QT container atom; it is not declared in the header files. You can create it with `QTNewAtomContainer` and insert it with `QTInsertChild`, using the following parameter:

Fields

`atomType`

Discussion

Constant `kSpriteImagesContainerAtomType`, designating atom type `'imct'`.

Parent Atom

`'dflt'` (page 24)
Parent atom can contain only one atom of this type.

Required Child Atoms

`'imag'` (page 38)
`Sprite` image atom. Any number allowed.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'imda' A sprite image data.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields

`atomType`

Discussion

Constant `kSpriteImageDataAtomType`, designating atom type `'imda'`.

`data`

Discussion
Image data.

Parent Atom

`'imag'` (page 38)
Parent atom can contain only one atom of this type.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'imgp' Panorama imaging parent atom.

This is a QT container atom; it is not declared in the header files. You can create it with `QTNewAtomContainer` and insert it with `QTInsertChild`, using the following parameter:

Fields
`atomType`

Discussion
Constant `kQTVRImagingParentAtomType`, designating atom type `'imgp'`.

Required Child Atoms

`'impn'` (page 41)
Panorama imaging atom. Any number allowed.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'imgr' A sprite image group ID atom.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields
`atomType`

Discussion
Constant `kSpriteImageGroupIDAtomType`, designating atom type `'imgr'`.

`data`

Discussion
The group ID, of type long.

Parent Atom

`'imag'` (page 38)
Parent atom can contain only one atom of this type.

Discussion
Each image in a sprite media key frame sample is assigned to a group. Add an atom of this type as a child of the `'imag'` (page 38) atom and set its leaf data to a long containing the group ID. For example, if the sample contains ten images where the first two images are equivalent, and the last eight images are equivalent, then you could assign a group ID of 1000 to the first two images, and a group ID of 1001 to the last eight images. This divides the images in the sample into two sets. The actual ID does not matter, it just needs to be a unique positive integer. Note that you must assign group IDs to your sprite sample if you want a sprite to display images with non-equivalent image descriptions (i.e., images with different dimensions).

Special Considerations

Although QuickTime does not currently use this atom internally, tools that edit sprite media can use the information provided to optimize certain operations, such as cut, copy, and paste.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'impn' Panorama imaging atom.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields

`atomType`

Discussion
Constant `kQTVRPanoImagingAtomType`, designating atom type `'impn'`.

`data`

Discussion
A `QTVRPanoImagingAtom` structure.

Parent Atom

`'imgp'` (page 40)
Parent atom can contain any number of atoms of this type.

Discussion
A `QTVRPanoImagingAtom` describes the default imaging characteristics for all the panoramic nodes in a scene. This atom overrides QuickTime VR's own defaults.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'imre' A sprite image data reference atom.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields

`atomType`

Discussion
Constant `kSpriteImageDataRefAtomType`, designating atom type `'imre'`.

`data`

Discussion
The data reference, which is similar to the `dataRef` parameter of `GetDataHandler`.

Parent Atom

`'imag'` (page 38)
Parent atom can contain only one atom of this type.

Discussion
You use the this atom type to specify that an image is referenced and how to access it. Add this atom as a child of the `'imag'` (page 38) atom instead of an `'imda'` (page 39) atom. Its ID should be 1.

Version Notes
Added in QuickTime 4.

See Also

See the `GetDataHandler` function. For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'imrg' Custom sprite image registration point atom.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields

`atomType`

Discussion
Constant `kSpriteImageRegistrationAtomType`, designating atom type `'imrg'`.

`data`

Discussion
The desired sprite registration point, a `FixedPoint` structure.

Parent Atom

`'imag'` (page 38)
Parent atom can contain only one atom of this type.

Discussion
`Sprite` images have a default registration point of 0, 0. To specify a different point, add an atom of this type as a child atom of the `'imag'` (page 38) and set its leaf data to a `FixedPoint` value with the desired registration point.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'imrt' A sprite image data reference type atom.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields

`atomType`

Discussion
Constant `kSpriteImageDataRefTypeAtomType`, designating atom type `'imrt'`.

`data`

Discussion
The data reference type, which is similar to the `dataRefType` parameter of `GetDataHandler`.

Parent Atom

`'imag'` (page 38)
Parent atom can contain only one atom of this type.

Discussion
You use the this atom type to specify that an image is referenced and how to access it. Add this atom as a child of the `'imag'` (page 38) atom. Its ID should be 1.

Version Notes
Added in QuickTime 4.

See Also

See the `GetDataHandler` function. For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

' in' `Track` input atom.

This is a QT container atom; it is not declared in the header files. You can create it with `QTNewAtomContainer` and insert it with `QTInsertChild`, using the following parameter:

Fields
`atomType`

Discussion
Value `' in'`; the first two characters are spaces.

Parent Atom

`'imap'` (page 39)
Parent atom can contain only one atom of this type.

Required Child Atoms

`' ty'` (page 83)
Input atom type. Only one allowed.

Optional Child Atoms

`'obid'` (page 56)
Object ID atom. Only one allowed.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'kmat' Defines a matte for a track's compressed media.

```
struct MatteCompressedAtom {
    long                size;
    long                atomType;
    long                flags;
    ImageDescription    matteImageDescription;
    char                matteData[1];
};
```

Fields
`size`

Discussion
The size in bytes of this atom structure.

`atomType`

Discussion
Constant `MatteCompAID`, designating atom type `'kmat'`.

`flags`

Discussion
One byte of version information followed by three bytes of flags. The flags bytes are not currently used.

`matteImageDescription`

Discussion
An `ImageDescription` data structure for the matte.

`matteData`

Discussion
An array of matte data.

Programming Info
C interface file: `MoviesFormat.h`

See Also

For the structure that contains this atom, see the `'matt'` (page 45) atom. For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'link' Contains specific information about a link hot spot.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields
`atomType`

Discussion
Constant `kQTVRLinkInfoAtomType`, designating atom type `'link'`.

`data`

Discussion
Link hot spot information.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'load' Contains preloading information for a track.

```
struct TrackLoadSettingsAtom {
    long                size;
    long                atomType;
    TrackLoadSettings   settings;
};
```

Fields
`size`

Discussion
The size in bytes of this atom structure.

`atomType`

Discussion
Constant `LoadSettingsAID`, designating atom type `'load'`.

`settings`

Discussion
A `TrackLoadSettings` data structure, which contains the actual data for this atom.

Programming Info
C interface file: `MoviesFormat.h`

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'LOOP' User data list entry atom: looping `style`.

```
struct MoviesUserData {
    long    size;
    long    udType;
    char    data[1];
};
```

Fields
`size`

Discussion
The size in bytes of this atom structure.

`udType`

Discussion
Value is `'LOOP'`.

`data`

Discussion
A long integer, indicating looping `style`: 0 for normal looping, 1 for palindromic looping.

Parent Atom
`'udta'` (page 84)
Parent atom can contain only one atom of this type.

Discussion
This atom is present only if the movie is set to loop.

Programming Info
C interface file: `MoviesFormat.h`

See Also

See the `MoviesUserData` structure. For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'matt' Defines a matte for a track's media.

```
struct MatteAtom {
    long                    size;
    long                    atomType;
    MatteCompressedAtom     aCompressedMatte;
};
```

Fields
`size`

Discussion
The size in bytes of this atom structure.

`atomType`

Discussion
Constant `MatteAID`, designating atom type `'matt'`.

`aCompressedMatte`

Discussion
A `'kmat'` (page 43) atom.

Required Child Atoms

`'kmat'` (page 43)
Only one allowed.

Programming Info

C interface file: `MoviesFormat.h`

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'maxr' Maximum data rate.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields

`atomType`

Discussion
Value is `'maxr'`.

`data`

Discussion
8 bytes.

Parent Atom

`'hinf'` (page 34)
Parent atom can contain only one atom of this type.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'mdat' Media data atom.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields

`atomType`

Discussion
Constant `MovieDataAtomType`, designating atom type `'mdat'`.

`data`

Discussion
Media data.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'mdhd' Specifies the characteristics of a media.

```
struct MediaHeaderAtom {
    long           size;
    long           atomType;
    MediaHeader    header;
};
```

Fields
`size`

Discussion
The size in bytes of this atom structure.

`atomType`

Discussion
Constant `MediaHeaderAID`, designating atom type `'mdhd'`.

`header`

Discussion
A `MediaHeader` data structure, which contains the actual data for this atom.

Parent Atom
`'mdia'` (page 47)
Parent atom can contain only one atom of this type.

Programming Info
C interface file: `MoviesFormat.h`

See Also
For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'mdia' Defines the media for a movie track.

```
struct MediaDirectory {
    long               size;
    long               atomType;
    MediaHeaderAtom    mediaHeader;
    HandlerAtom        mediaHandler;
    MediaInfo          mediaInfo;
};
```

Fields
`size`

Discussion
The size in bytes of this atom structure.

`atomType`

Discussion
Constant `MediaAID`, designating atom type `'mdia'`.

`mediaHeader`

Discussion
A `'mdhd'` (page 47) atom that specifies general characteristics of the media.

`mediaHandler`

Discussion
A `'hdlr'` (page 33) atom that defines a handler for the media.

`mediaInfo`

Discussion
A `'minf'[generic]` (page 49) atom structure that contains data to be passed to the media handler.

Parent Atom

`'trak'` (page 79)
Parent atom can contain only one atom of this type.

Required Child Atoms

`'mdhd'` (page 47)
General characteristics of the media. Only one allowed.

Optional Child Atoms

`'hdlr'` (page 33)
The type of media this atom contains. Only one allowed.
`'minf'[generic]` (page 49)
Data that is specific to a media handler. Only one allowed.
`'udta'` (page 84)
User data atom. Only one allowed.

Discussion
The `'hdlr'` atom specifies what type of media this atom contains; for example, video or sound. The content of the `'minf'[generic]` atom is specific to the media handler that is to interpret the media.

Programming Info
C interface file: `MoviesFormat.h`

See Also

See the `MediaHeader` structure. For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'mime'[atom] Defines a graphics export MIME type.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields
`atomType`

Discussion
Constant `kGraphicsExportMIMEType`, designating atom type `'mime'`.

`data`

Discussion
A nonterminated string containing a MIME type.

Parent Atom

`'expo'` (page 30)
Parent atom can contain only one atom of this type.

See Also

See the `GraphicsImportGetExportImageTypeList`, `GraphicsImportGetMIMETypeList`, and `GraphicsExportGetMIMETypeList` functions. For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'minf'[base] Provides data that is specific to a handler for media other than video or sound.

```
struct MediaInfo {
    long    size;
    long    atomType;
};
```

Fields
`size`

Discussion
The size in bytes of this atom structure.

`atomType`

Discussion
Constant `MediaInfoAID`, designating atom type `'minf'`.

Parent Atom

`'mdia'` (page 47)
Parent atom can contain only one atom of this type.

Required Child Atoms

`'gmhd'` (page 32)
Generic media information atom. Only one allowed.
`'gmin'` (page 33)
Provides data that is specific to a handler for media other than video or sound. Only one allowed.

Discussion
Media information atoms store handler-specific information for the media data that constitutes a track. The media handler uses this information to map from media time to media data. The format and content of media information atoms are dictated by the media handler that is responsible for interpreting the media data stream. Another media handler would not know how to interpret this information.

Programming Info
C interface file: `MoviesFormat.h`

See Also

This isotope of the `'minf'` atom provides data that is specific to a handler for media other than video or sound. Handler-specific data for sound and video are provided by the `'minf'[sound]` (page 50) atom and the `'minf'[video]` (page 51) atom. For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'minf'[generic] Provides data that is specific to a handler for media other than video or sound.

```
struct MediaInfo {
    long    size;
    long    atomType;
};
```

Fields
`size`

Discussion
The size in bytes of this atom structure.

`atomType`

Discussion
Constant `MediaInfoAID`, designating atom type `'minf'`.

Parent Atom

'mdia' (page 47)
Parent atom can contain only one atom of this type.

Required Child Atoms

'gmhd' (page 32)
Generic media information atom. Only one allowed.
'hdlr' (page 33)
The type of media this atom contains. Only one allowed.

Optional Child Atoms

'dinf' (page 25)
Specifies where media data is stored. Only one allowed.
'stbl' (page 68)
Contains information for converting from media time to sample number to sample location and indicates how to interpret samples and chunks. Only one allowed.

Discussion

Media information atoms store handler-specific information for the media data that constitutes a track. The media handler uses this information to map from media time to media data. The format and content of media information atoms are dictated by the media handler that is responsible for interpreting the media data stream. Another media handler would not know how to interpret this information.

Programming Info

C interface file: MoviesFormat.h

See Also

This isotope of the 'minf' atom provides data that is specific to a handler for media other than video or sound. Handler-specific data for sound and video are provided by the 'minf'[sound] (page 50) atom and the 'minf'[video] (page 51) atom. For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'minf'[sound] Sound media information atom.

```
struct MediaInfo {
    long    size;
    long    atomType;
};
```

Fields

size

Discussion

The size in bytes of this atom structure.

atomType

Discussion

Constant MediaInfoAID, designating atom type 'minf'.

Parent Atom

'mdia' (page 47)
Parent atom can contain only one atom of this type.

Required Child Atoms

'smhd' (page 65)
Contains sound stereo balance information. Only one allowed.

'hdlr' (page 33)
The type of media this atom contains. Only one allowed.

Optional Child Atoms

'dinf' (page 25)
Specifies where media data is stored. Only one allowed.
'stbl' (page 68)
Contains information for converting from media time to sample number to sample location and indicates how to interpret samples and chunks. Only one allowed.

Discussion
Media information atoms store handler-specific information for the media data that constitutes a track. The media handler uses this information to map from media time to media data. The format and content of media information atoms are dictated by the media handler that is responsible for interpreting the media data stream. Another media handler would not know how to interpret this information.

Programming Info
C interface file: MoviesFormat.h

See Also

This isotope of the 'minf' atom provides handler-specific data for sound. Handler-specific data for video is provided by the the 'minf'[video] (page 51) atom. For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'minf'[video] Video media information atom.

```
struct MediaInfo {
    long    size;
    long    atomType;
};
```

Fields
size

Discussion
The size in bytes of this atom structure.

atomType

Discussion
Constant MediaInfoAID, designating atom type 'minf'.

Parent Atom

'mdia' (page 47)
Parent atom can contain only one atom of this type.

Required Child Atoms

'vmhd'[media] (page 86)
Stores handler-specific information for video media in a track. Only one allowed.
'hdlr' (page 33)
The type of media this atom contains. Only one allowed.

Optional Child Atoms

'dinf' (page 25)
Specifies where media data is stored. Only one allowed.
'stbl' (page 68)
Contains information for converting from media time to sample number to sample location and indicates how to interpret samples and chunks. Only one allowed.

Discussion
Media information atoms store handler-specific information for the media data that constitutes a track. The media handler uses this information to map from media time to media data. The format and content of media information atoms are dictated by the media handler that is responsible for interpreting the media data stream. Another media handler would not know how to interpret this information.

Programming Info
C interface file: `MoviesFormat.h`

See Also

This isotope of the `'minf'` atom provides handler-specific data for video. Handler-specific data for sound is provided by the `'minf'[sound]` (page 50) atom. For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'moov' Contains the top-level atoms that constitute a movie.

```
struct MovieDirectory {
    long                    size;
    long                    atomType;
    MovieHeaderAtom         header;
    ClippingAtom            movieClip;
    TrackDirectoryEntry     track[1];
    UserDataAtom            userData;
};
```

Fields
`size`

Discussion
The size in bytes of this atom structure.

`atomType`

Discussion
Constant `MovieAID`, designating atom type `'moov'`.

`header`

Discussion
A `'mvhd'` (page 53) atom that specifies the general characteristics of the movie.

`movieClip`

Discussion
A `'clip'` (page 11) atom that defines the clipping region for the movie.

`track`

Discussion
An array of one or more `TrackDirectoryEntry` data structures, each of which includes a `'trak'` (page 79) atom that defines a track in the movie.

`userData`

Discussion
A `'udta'` (page 84) atom, which contains user data.

Required Child Atoms

`'mvhd'` (page 53)
Specifies the general characteristics of a movie. Only one allowed.

Optional Child Atoms

`'clip'` (page 11)
Defines the clipping region for the movie. Only one allowed.
`'trak'` (page 79)
Defines a single track of the movie. Any number allowed.
`'udta'` (page 84)
User data atom. Only one allowed.
`'ctab'` (page 21)
Color table atom. Only one allowed.
`'ptv '` (page 57)
Defines a movie's full screen mode.

Discussion

You use movie atoms to specify the information that defines a movie; that is, the information that allows your application to understand the data that is stored in the movie data atom. The movie atom contains the movie header atom, which defines the time scale and duration information for the entire movie, as well as its display characteristics. In addition, the movie atom contains each track in the movie.

Programming Info

C interface file: `MoviesFormat.h`

See Also

See the `MovieHeader` structure. For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'mvhd' Specifies the characteristics of an entire movie.

```
struct MovieHeaderAtom {
    long           size;
    long           atomType;
    MovieHeader    header;
};
```

Fields

`size`

Discussion

The size in bytes of this atom structure.

`atomType`

Discussion

Constant `MovieHeaderAID`, designating atom type `'mvhd'`.

`header`

Discussion

A `MovieHeader` data structure, which contains the actual data for this atom.

Parent Atom

`'moov'` (page 52)
Parent atom can contain only one atom of this type.

Discussion

You use the movie header atom to specify the characteristics of an entire QuickTime movie. The data contained in this atom defines characteristics of the entire QuickTime movie, such as time scale and duration.

Programming Info

C interface file: `MoviesFormat.h`

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'name'[sprite] A sprite name atom.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields
`atomType`

Discussion
Constant `kSpriteNameAtomType`, designating atom type `'name'`.

`data`

Discussion
One or more ASCII characters comprising the sprite's name.

Parent Atom

`'sprt'` (page 66)
Parent atom can contain only one atom of this type.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'name'[userdata] User data list entry atom: name of object.

```
struct MoviesUserData {
    long    size;
    long    udType;
    char    data[1];
};
```

Fields
`size`

Discussion
The size in bytes of this atom structure.

`udType`

Discussion
Constant `kUserDataName`, designating atom type `'name'`.

`data`

Discussion
A name string.

Parent Atom

`'udta'` (page 84)
Parent atom can contain only one atom of this type.

Programming Info
C interface file: `MoviesFormat.h`

See Also

See the `MoviesUserData` structure. For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'ndhd' Node header atom.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields
`atomType`

Discussion
Constant `kQTVRNodeHeaderAtomType`, designating atom type `'ndhd'`.

`data`

Discussion
Node header information.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'nloc' QTVR node location atom.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields
`atomType`

Discussion
Constant `kQTVRNodeLocationAtomType`, designating atom type `'nloc'`.

`data`

Discussion
Node location.

Parent Atom

`'vrni'` (page 88)
Parent atom can contain only one atom of this type.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'nump' Total number of network packets that will be sent.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields
`atomType`

Discussion
Value is `'nump'`.

`data`

Discussion
8 bytes.

Parent Atom

`'hinf'` (page 34)
Parent atom can contain only one atom of this type.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'obid' Object ID atom.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields
`atomType`
Discussion
Constant `kTrackModifierObjectID`, designating atom type `'obid'`.

`data`
Discussion
The object ID.

Parent Atom

`' in'` (page 43)
Parent atom can contain only one atom of this type.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'payt' Payload type atom.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields
`atomType`
Discussion
Value is `'payt'`.

`data`
Discussion
Payload type, which includes payload number (32-bits) followed by an RTP map payload string (a Pascal string).

Parent Atom

`'hinf'` (page 34)
Parent atom can contain only one atom of this type.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'pdat' Panorama sample atom.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields
`atomType`
Discussion
Constant `kQTVRPanoSampleDataAtomType`, designating atom type `'pdat'`.

`data`

Discussion
A panorama sample.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'pmax' Largest packet, in bytes; includes 12-byte RTP header.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields
`atomType`

Discussion
Value is `'pmax'`.

`data`

Discussion
4 bytes.

Parent Atom

`'hinf'` (page 34)
Parent atom can contain only one atom of this type.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'pnot' Reference to movie preview data.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields
`atomType`

Discussion
Constant `ShowFilePreviewComponentType`, designating atom type `'pnot'`.

`data`

Discussion
Reference to a movie preview.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'ptv ' Defines a movie's full screen mode.

This is a classic atom; you can access its information by calculating offsets.

Fields
`size`

Discussion
Value is 0x0000000C.

`atomType`

Discussion
Value is `'ptv '`.

Parent Atom
`'moov'` (page 52)
Contains the top-level atoms that constitute a movie.

Data offsets
0x0000
Display size: a 16-bit big-endian integer (see below) indicating the display size for the movie.
0x0002
Reserved: set to 0.
0x0004
Reserved: set to 0.
0x0006
Slide show: an 8-bit Boolean whose value is 1 for a slide show. In slide show mode, the movie advances one frame each time the right-arrow key is pressed. Audio is muted.
0x0007
Play on open: an 8-bit boolean whose value is normally 1, indicating that the movie should play when opened. Since there is no visible controller in full-screen mode, applications should always set this field to 1 to prevent user confusion.

Display size constants
0x00000
The movie should be played at its normal size.
0x0001
The movie should be played at double size.
0x0002
The movie should be played at half size.
0x0003
The movie should be scaled to fill the screen.
0x0004
The movie should be played at its current size. This value is normally used when the `'ptv '` atom is inserted transiently and the movie has been temporarily resized.

See Also
For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'qdrg' QuickDraw region atom.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields
`atomType`

Discussion
Constant `kTweenRegionData`, designating atom type `'qdrg'`.

`data`

Discussion
Two `Rect` structures and a `MacRegion` structure.

Discussion
This atom's ID must be 1.

See Also

See the `TweenerInitialize` function. For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'rdrf' Provides a reference to an alternate movie.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields
`atomType`

Discussion
Constant `ReferenceMovieDataRefAID`, designating atom type `'rdrf'`.

`data`

Discussion
A `ReferenceMovieDataRefRecord` data structure. The alternate movie referenced by this structure is the movie associated with the parent `'rmda'` (page 60) atom.

Parent Atom

`'rmda'` (page 60)
Parent atom can contain only one atom of this type.

Discussion
Alias data references are the contents of `AliasRecord` structures. The QuickTime plug-in is smart enough to convert a relative alias to a relative URL. To designate the anchor file for a relative alias, pass the `FSSpec` structure that specifies the file you are creating. You can pass absolute or relative URLs; if the movie is loaded from the desktop, QuickTime will convert a relative URL into a relative alias.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'reso' `Pixmap` resolution atom.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields
`atomType`

Discussion
Constant `kQTResolutionSettings`, designating atom type `'reso'`.

`data`

Discussion
A `QTResolutionSettings` structure.

Parent Atom

`'vide'` (page 86)
Parent atom can contain only one atom of this type.

Discussion
This atom specifies the resolution for the `PixMap` structure passed to the compressor.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'rmcd' Provides component availability information for selecting an alternate movie.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields
`atomType`
Discussion
Constant `ReferenceMovieComponentCheckAID`, designating atom type `'rmcd'`.

`data`
Discussion
A `QTAltComponentCheckRecord` data structure.

Parent Atom
`'rmda'` (page 60)
Parent atom can contain any number of atoms of this type.

See Also
See the `QTAltComponentCheckRecord` structure. For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'rmcs' Provides CPU speed information for selecting an alternate movie.

This is a QT atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields
`atomType`
Discussion
Constant `ReferenceMovieCPURatingAID`, designating atom type `'rmcs'`.

`data`
Discussion
A `QTAltCPURatingRecord` data structure.

Parent Atom
`'rmda'` (page 60)
Parent atom can contain only one atom of this type.

See Also
See the `QTAltCPURatingRecord` structure. For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'rmda' Provides criteria for selecting an alternate movie.

This is a QT atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameter:

Fields
`atomType`
Discussion
Constant `ReferenceMovieDescriptorAID`, designating atom type `'rmda'`.

Parent Atom

`'rmra'` (page 62)
Parent atom can contain only one atom of this type.

Required Child Atoms

`'rdrf'` (page 59)
A reference to an alternate movie. Only one allowed.

Optional Child Atoms

`'rmdr'` (page 61)
Data rate information for selecting an alternate movie. Only one allowed.
`'rmvc'` (page 63)
Version criteria for selecting an alternate movie. Multiples allowed.
`'rmcd'` (page 60)
Component availability information for selecting an alternate movie. Multiples allowed.
`'rmqu'` (page 62)
Playback quality information for selecting an alternate movie. Only one allowed.
`'rmla'` (page 61)
Language information for selecting an alternate movie. Only one allowed.
`'rmcs'` (page 60)
CPU speed information for selecting an alternate movie. Only one allowed.

Discussion

The `'rdrf'` atom contains a `ReferenceMovieDataRefRecord`, which designates an alternate movie. The `'rmda'` atom's optional atoms help QuickTime decide whether or not to run that movie. If multiple `'rmvc'` or `'rmcd'` atoms are present, all their criteria must be satisfied for the movie to play.

See Also

See the `ReferenceMovieDataRefRecord` structure. For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'rmdr' Provides data rate information for selecting an alternate movie.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields

`atomType`

Discussion

Constant `ReferenceMovieDataRateAID`, designating atom type `'rmdr'`.

`data`

Discussion

A `QTAltDataRateRecord` data structure.

Parent Atom

`'rmda'` (page 60)
Parent atom can contain only one atom of this type.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'rmla' Provides language information for selecting an alternate movie.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields
`atomType`

Discussion
Constant `ReferenceMovieLanguageAID`, designating atom type `'rmla'`.

`data`

Discussion
A `QTAltLanguageRecord` structure.

Parent Atom

`'rmda'` (page 60)
Parent atom can contain only one atom of this type.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'rmqu' Provides playback quality information for selecting an alternate movie.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields
`atomType`

Discussion
Constant `ReferenceMovieQualityAID`, designating atom type `'rmqu'`.

`data`

Discussion
A quality value of type SInt32. Higher quality values are selected over lower quality values.

Parent Atom

`'rmda'` (page 60)
Parent atom can contain only one atom of this type.

Discussion
If the criteria established by the `'rmdr'` (page 61), `'rmvc'` (page 63), and `'rmcd'` (page 60) atoms are equally satisfied by two or more alternate movies, the one with the highest quality value will be selected.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'rmra' Designates a reference movie.

This is a QT atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameter:

Fields
`atomType`

Discussion
Constant `ReferenceMovieRecordAID`, designating atom type `'rmra'`.

Parent Atom

`'moov'` (page 52)
Parent atom can contain only one atom of this type.

Required Child Atoms

`'rmda'` (page 60)
Provides criteria for selecting an alternate movie. Only one allowed.

Discussion
You insert an `'rmra'` atom in a `'moov'` atom to create a reference movie. Each `'rmda'` atom in the `'rmra'` atom designates an alternate movie.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'rmvc' Provides version criteria for selecting an alternate movie.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields
`atomType`

Discussion
Constant `ReferenceMovieVersionCheckAID`, designating atom type `'rmvc'`.

`data`

Discussion
A `QTAltVersionCheckRecord` data structure.

Parent Atom

`'rmda'` (page 60)
Parent atom can contain any number of atoms of this type.

Discussion
This optional atom in a `'rmda'` atom lets you demand minimum product version criteria for selecting an alternate movie. For example, a movie that needs QuickTime VR 2.1 or later could require a Gestalt `'qtvv'` value of 0x02100000 or higher.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'scpt' Transcript track reference type atom.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields
`atomType`

Discussion
Value is `'scpt'`.

`data`

Discussion
A list of track ID values (32-bit integers) specifying the related tracks. Note that a track ID value can be set to 0 to indicate an unused entry in the atom. Doing this can be more convenient than deleting the reference.

Parent Atom

`'tref'` (page 80)
Parent atom can contain only one atom of this type.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'sdp ' SDP text for a hint track.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields
`atomType`

Discussion
Value is `'sdp '`. The fourth character is a space.

`data`

Discussion
SDP text.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'sean' The outermost atom container, of which all other atoms are children.

This is a QT container atom; it is not declared in the header files. You can create it with `QTNewAtomContainer`, using the following parameter:

Fields
`atomData`

Discussion
A pointer to a memory location that will hold the new atom.

Discussion
After creating a `'sean'` atom, you can populate it with other atoms by using `QTInsertChild`.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'SelO' User data list entry atom: play selection only.

```
struct MoviesUserData {
    long    size;
    long    udType;
    char    data[1];
};
```

Fields
`size`

Discussion
The size in bytes of this atom structure.

`udType`

Discussion
Constant `'SelO'`.

`data`

Discussion
A byte indicating that only the selected area of the movie should be played.

Parent Atom

`'udta'` (page 84)

Parent atom can contain only one atom of this type.

Programming Info

C interface file: `MoviesFormat.h`

See Also

See the `MoviesUserData` function. For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'skip' Unused space available in the file.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields

`atomType`

Discussion

Constant `SkipAtomType`, designating atom type `'skip'`.

`data`

Discussion

Any number of bytes of free space.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'smhd' Contains sound stereo balance information.

```
struct SoundMediaInfoHeaderAtom {
    long                    size;
    long                    atomType;
    SoundMediaInfoHeader    smiHeader;
};
```

Fields

`size`

Discussion

The size in bytes of this atom structure.

`atomType`

Discussion

Constant `SoundMediaInfoHeaderAID`, designating atom type `'smhd'`.

`smiHeader`

Discussion

A `SoundMediaInfoHeader` data structure, which contains the actual data for this atom.

Discussion

The `SoundMediaInfoHeader` data structure currently contains only stereo balance information.

Programming Info

C interface file: `MoviesFormat.h`

See Also

For the structure that contains this atom, see `'minf'[sound]` (page 50). For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'sprt' A key frame sprite definition.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields

`atomType`

Discussion
Constant `kSpriteAtomType`, designating atom type `'sprt'`.

`data`

Discussion
A list of sprite property constants (see below).

Sprite property constants

`kSpritePropertyMatrix`
(Value is 1). Describes the sprite's location and scaling within its sprite world or sprite track. By modifying a sprite's matrix, you can modify the sprite's location so that it appears to move in a smooth path on the screen or so that it jumps from one place to another. You can modify a sprite's size, so that it shrinks, grows, or stretches. Depending on which image compressor is used to create the sprite images, other transformations, such as rotation, may be supported as well. Translation-only matrixes provide the best performance.
`kSpritePropertyVisible`
(Value is 4). Specifies whether or not the sprite is visible. To make a sprite visible, you set the sprite's visible property to true.
`kSpritePropertyLayer`
(Value is 5). Contains a 16-bit integer value specifying the layer into which the sprite is to be drawn. Sprites with lower layer numbers appear in front of sprites with higher layer numbers. To designate a sprite as a background sprite, you should assign it the special layer number `kBackgroundSpriteLayerNum`.
`kSpritePropertyGraphicsMode`
(Value is 6). Specifies a graphics mode and blend color that indicates how to blend a sprite with any sprites behind it and with the background. To set a sprite's graphics mode, you call `SetSpriteProperty`, passing a pointer to a `ModifierTrackGraphicsModeRecord` structure.
`kSpritePropertyActionHandlingSpriteID`
(Value is 8). Specifies another sprite by ID that delegates QT events.
`kSpritePropertyImageIndex`
(Value is 100). Contains the atom ID of the sprite's image atom.
`kSpriteUsesImageIDsAtomType`
(Value is `'uses'`). Lets a sprite specify the subset of images that `kSpritePropertyImageIndex` can refer to.

Parent Atom

`'stss'` (page 73)
Parent atom can contain only one atom of this type.

Discussion
`Sprite` atoms should have ID numbers start at 1 and count consecutively upward.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'sptl' Specifies which graphics export compressor to use, its depth, and the spatial quality.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields
`size`
Discussion
The size in bytes of this atom structure.

`atomType`
Discussion
Constant `scSpatialSettingsType`, designating atom type `'sptl'`.

`data`
Discussion
A pointer to `SCSpatialSettings` structure.

See Also

See the `GraphicsExportCanUseCompressor` function. For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'ssrc' Nonprimary source track reference type atom.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields
`atomType`
Discussion
Constant `kTrackModifierReference`, designating atom type `'ssrc'`.

`data`
Discussion
A list of track ID values (32-bit integers) specifying the related tracks. Note that a track ID value can be set to 0 to indicate an unused entry in the atom. Doing this can be more convenient than deleting the reference.

Parent Atom

`'tref'` (page 80)
Parent atom can contain only one atom of this type.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'sstr' Specifies the ID of a string variable, contained in a sprite track, to display in the status area of the browser.

This is a QT container atom; it is not declared in the header files. You can create it with `QTNewAtomContainer` and insert it with `QTInsertChild`, using the following parameter:

Fields
`atomType`
Discussion
Value is `'sstr'`.

Parent Atom

'beha' (page 11)

Defines sprite behavior.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'stbl' Contains information for converting from media time to sample number to sample location and indicates how to interpret samples and chunks.

```
struct SampleTableAtom {
    long                    size;
    long                    atomType;
    SampleDescriptionAtom   sampleDescription;
    TimeToSampleNumAtom     timeToSampleNum;
    SampleToChunkAtom       sampleToChunk;
    SyncSampleAtom          syncSample;
    SampleSizeAtom          sampleSize;
    ChunkOffsetAtom         chunkOffset;
    ShadowSyncAtom          shadowSync;
};
```

Fields

`size`

Discussion

The size in bytes of this atom structure.

`atomType`

Discussion

Constant `SampleTableAID`, designating atom type `'stbl'`.

`sampleDescription`

Discussion

A 'stsd' (page 71) atom, which contains information required to decode the samples in the media.

`timeToSampleNum`

Discussion

A 'stts' (page 75) atom that relates sample numbers to sample durations.

`sampleToChunk`

Discussion

A 'stsc' (page 71) atom that maps sample numbers to chunk numbers.

`syncSample`

Discussion

A 'stss' (page 73) atom that identifies the key frames in the media.

`sampleSize`

Discussion

A 'stsz' (page 74) atom, which identifies the size of each sample in the media.

`chunkOffset`

Discussion

A 'stco' (page 69) atom that identifies the location of each data chunk in the media.

`shadowSync`

Discussion
A `'stsh'` (page 72) atom, which lists self-contained sync samples that are alternates for existing frame difference samples. This field may be omitted.

Discussion
This atom contains the information you need to find a sample number based on a time and to find the sample's location based on the sample number. Samples are organized into chunks, containing one or more samples. This atom contains the information you need to find out which chunk holds a given sample, where that chunk begins, and where in that chunk you can find the sample.

Programming Info
C interface file: `MoviesFormat.h`

See Also

For the atoms that may contain this atom, see `'minf'[generic]` (page 49), `'minf'[sound]` (page 50), and `'minf'[video]` (page 51). For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'stco' Identifies the location of each chunk of data in the media's data stream.

```
struct ChunkOffsetAtom {
    long    size;
    long    atomType;
    long    flags;
    long    numEntries;
    long    chunkOffsetTable[1];
};
```

Fields
`size`

Discussion
The size in bytes of this atom structure.

`atomType`

Discussion
Constant `STChunkOffsetAID`, designating atom type `'stco'`.

`flags`

Discussion
One byte of version information followed by three bytes of flags. The flags bytes are not currently used.

`numEntries`

Discussion
The number of entries in `chunkOffsetTable`.

`chunkOffsetTable`

Discussion
An array of chunk offset values.

Discussion
A chunk is a collection of data samples in a media that allows optimized data access. A chunk may contain one or more samples. Chunks in a media may have different sizes, and the samples within a chunk may have different sizes.

Programming Info
C interface file: `MoviesFormat.h`

See Also

For the structure that contains this atom, see `'stbl'` (page 68). For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'strt' Defines the starting offset of hypertext in a text stream.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields
`atomType`
Discussion
Value is `'strt'`.

`data`
Discussion
The starting offset of hypertext.

Parent Atom

`'htxt'` (page 37)
Parent atom can contain only one atom of this type.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'strv' Contains a string variable for a sprite.

This is a QT container atom; it is not declared in the header files. You can create it with `QTNewAtomContainer` and insert it with `QTInsertChild`, using the following parameter:

Fields
`atomType`
Discussion
Value is `'strv'`.

Parent Atom

`'vars'` (page 86)
Contains variables for a sprite.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'stsc' Maps sample numbers to chunk numbers.

```
struct SampleToChunkAtom {
    long             size;
    long             atomType;
    long             flags;
    long             numEntries;
    SampleToChunk    sampleToChunkTable[1];
};
```

Fields
`size`

Discussion
The size in bytes of this atom structure.

`atomType`

Discussion
Constant `STSampleToChunkAID`, designating atom type `'stsc'`.

`flags`

Discussion
One byte of version information followed by three bytes of flags. The flags bytes are not currently used.

`numEntries`

Discussion
The number of entries in `sampleToChunkTable`.

`sampleToChunkTable`

Discussion
An array of `SampleToChunk` data structures, which contain the actual data for this atom.

Discussion
A chunk is a collection of data samples in a media that allows optimized data access. A chunk may contain one or more samples. Chunks in a media may have different sizes, and the samples within a chunk may have different sizes.

Programming Info
C interface file: `MoviesFormat.h`

See Also

For the structure that contains this atom, see `'stbl'` (page 68). For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'stsd' Holds one or more sample description structures.

```
struct SampleDescriptionAtom {
    long                size;
    long                atomType;
    long                flags;
    long                numEntries;
    SampleDescription   sampleDescTable[1];
};
```

Fields
`size`

Discussion
The size in bytes of this atom structure.

`atomType`

Discussion
Constant `STSampleDescAID`, designating atom type `'stsd'`.

`flags`

Discussion
One byte of version information followed by three bytes of flags. The flags bytes are not currently used.

`numEntries`

Discussion
Number of entries in `sampleDescTable`.

`sampleDescTable`

Discussion
An array of `SampleDescription` data structures, which contain the actual data for this atom.

Discussion
In QuickTime streaming, this atom describes the data format of the hint samples and contains track-level information, such as the RTP timescale for a track.

Programming Info
C interface file: `MoviesFormat.h`

See Also

For the structure that contains this atom, see `'stbl'` (page 68). For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'stsh' Lists self-contained sync samples that are alternates for existing frame difference samples.

```
struct ShadowSyncAtom {
    long          size;
    long          atomType;
    long          flags;
    long          numEntries;
    ShadowSync    shadowSyncTable[1];
};
```

Fields
`size`

Discussion
The size in bytes of this atom structure.

`atomType`

Discussion
Constant `STShadowSyncAID`, designating atom type `'stsh'`.

`flags`

Discussion
One byte of version information followed by three bytes of flags. The flags bytes are not currently used.

`numEntries`

Discussion
The number of entries in `shadowSyncTable`.

`shadowSyncTable`

Discussion
An array of `ShadowSync` data structures, which contain the actual data for this atom.

Discussion
Shadow sync atoms are used to optimize random access operations on a movie, thereby enhancing playback performance.

Programming Info
C interface file: `MoviesFormat.h`

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'stss' Identifies the key frames in a media.

```
struct SyncSampleAtom {
    long    size;
    long    atomType;
    long    flags;
    long    numEntries;
    long    syncSampleTable[1];
};
```

Fields
`size`

Discussion
The size in bytes of this atom structure.

`atomType`

Discussion
Constant `STSyncSampleAID`, designating atom type `'stss'`.

`flags`

Discussion
Flags (currently not used).

`numEntries`

Discussion
The number of entries in `syncSampleTable`.

`syncSampleTable`

Discussion
An array of physical sample numbers, each of which identifies a key frame in the media.

Discussion
In a media that contains compressed data, key frames define starting points for portions of a temporally compressed sequence. Each key frame is independent of the preceding frames. Subsequent frames may depend on the key frame.

Programming Info
C interface file: `MoviesFormat.h`

See Also

For the structure that contains this atom, see `'stbl'` (page 68). For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'stsz' Identifies the size of each sample in a media.

```
struct SampleSizeAtom {
    long    size;
    long    atomType;
    long    flags;
    long    sampleSize;
    long    numEntries;
    long    sampleSizeTable[1];
};
```

Fields

`size`

Discussion
The size in bytes of this atom structure.

`atomType`

Discussion
Constant `STSampleSizeAID`, designating atom type `'stsz'`.

`flags`

Discussion
One byte of version information followed by three bytes of flags. The flags bytes are not currently used.

`sampleSize`

Discussion
The number of bytes in the samples. If all the samples are the same size, `sampleSize` indicates the size of all the samples. If `sampleSize` is set to 0, then the samples have different sizes, and those sizes are stored in `sampleSizeTable`.

`numEntries`

Discussion
The number of entries in `sampleSizeTable`.

`sampleSizeTable`

Discussion
An array of numbers, one for every sample. This field is indexed by sample number; the first entry corresponds to the first sample, the second to the second sample, and so on.

Programming Info

C interface file: `MoviesFormat.h`

See Also

For the structure that contains this atom, see `'stbl'` (page 68). For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'stts' Holds one or more structures that relate sample numbers to sample durations.

```
struct TimeToSampleNumAtom {
    long                size;
    long                atomType;
    long                flags;
    long                numEntries;
    TimeToSampleNum     timeToSampleNumTable[1];
};
```

Fields
`size`

Discussion
The size in bytes of this atom structure.

`atomType`

Discussion
Constant `STTimeToSampAID`, designating atom type `'stts'`.

`flags`

Discussion
One byte of version information followed by three bytes of flags. The flags bytes are not currently used.

`numEntries`

Discussion
The number of entries in `timeToSampleNumTable`.

`timeToSampleNumTable`

Discussion
An array of `TimeToSampleNum` data structures, each of which maps a sample number to its sample duration.

Discussion
Entries in `timeToSampleNumTable` collect samples according to their order in the media and their duration. If consecutive samples have the same duration, a single table entry may be used to define more than one sample. In these cases, the `count` field indicates the number of consecutive samples that have the same duration. For example, if a video media had a constant frame rate, `timeToSampleNumTable` would have one entry.

Programming Info
C interface file: `MoviesFormat.h`

See Also

For the structure that contains this atom, see `'stbl'` (page 68). For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'sync' Synchronization track reference type atom.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields
`atomType`

Discussion
Value is `'sync'`.

`data`

Discussion
A list of track ID values (32-bit integers) specifying the related tracks. Note that a track ID value can be set to 0 to indicate an unused entry in the atom. Doing this can be more convenient than deleting the reference.

Parent Atom
`'tref'` (page 80)
Parent atom can contain only one atom of this type.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'tbox' Defines a text box rectangle.

```
struct TextBoxAtom {
    long    size;
    long    atomType;
    Rect    textBox;
};
```

Fields
`size`

Discussion
The size in bytes of this atom structure.

`atomType`

Discussion
Value is `'tbox'`.

`textBox`

Discussion
A new text box rectangle, which overrides the rectangle defined by the `defaultTextBox` constant.

Discussion
This is a classic atom; you can access its information by calculating offsets.

Programming Info
C interface file: `MoviesFormat.h`

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'tcmi' Time code media information atom.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields
`atomType`

Discussion
Value is `'tcmi'`.

`data`

Discussion
Time code media information.

Parent Atom

`'minf'[video]` (page 51)

Parent atom can contain only one atom of this type.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'tkhd' Specifies the characteristics of a track in a movie.

```
struct TrackHeaderAtom {
    long           size;
    long           atomType;
    TrackHeader    header;
};
```

Fields

`size`

Discussion

The size in bytes of this atom structure.

`atomType`

Discussion

Constant `TrackHeaderAID`, designating atom type `'tkhd'`.

`header`

Discussion

A `TrackHeader` structure that contains the actual data for this atom.

Parent Atom

`'trak'` (page 79)

Parent atom can contain only one atom of this type.

Programming Info

C interface file: `MoviesFormat.h`

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'tmax' Largest relative transmission time, in milliseconds.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields

`atomType`

Discussion

Value is `'tmax'`.

`data`

Discussion

4 bytes.

Parent Atom

`'hinf'` (page 34)

Parent atom can contain only one atom of this type.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'tmcd' Time code track reference type atom.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields
`atomType`

Discussion
Constant `TimeCodeMediaType`, designating atom type `'tmcd'`.

`data`

Discussion
A list of track ID values (32-bit integers) specifying the related tracks. Note that a track ID value can be set to 0 to indicate an unused entry in the atom. Doing this can be more convenient than deleting the reference.

Parent Atom

`'tref'` (page 80)
Parent atom can contain only one atom of this type.

See Also

See the `TimeCodeDescription` structure. For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'tmin' Smallest relative transmission time, in milliseconds.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields
`atomType`

Discussion
Value is `'tmin'`.

`data`

Discussion
4 bytes.

Parent Atom

`'hinf'` (page 34)
Parent atom can contain only one atom of this type.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'tpyl' Total number of bytes that will be sent, not including 12-byte RTP headers.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields
`atomType`

Discussion
Value is `'tpyl'`.

`data`

Discussion
8 bytes.

Parent Atom

`'hinf'` (page 34)
Parent atom can contain only one atom of this type.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'trak' Defines a single track of a movie.

```
struct TrackDirectory {
    long                size;
    long                atomType;
    TrackHeaderAtom     trackHeader;
    ClippingAtom        trackClip;
    EditsAtom           edits;
    MediaDirectory      media;
    UserDataAtom        userData;
};
```

Fields
`size`

Discussion
The size in bytes of this atom structure.

`atomType`

Discussion
Constant `TrackAID`, designating atom type `'trak'`.

`trackHeader`

Discussion
A `'tkhd'` (page 77) atom, which specifies general characteristics of the track.

`trackClip`

Discussion
A `'clip'` (page 11) atom, which defines the track's clipping region.

`edits`

Discussion
An `'edts'` (page 27) atom, which defines the portions of the track's media that are going into the track.

`media`

Discussion
A `'mdia'` (page 47) atom structure that defines the media for the track.

`userData`

Discussion
A `'udta'` (page 84) atom that contains user data.

Parent Atom

`'moov'` (page 52)
Parent atom can contain any number of atoms of this type.

Required Child Atoms

`'tkhd'` (page 77)
Specifies general characteristics of the track. Only one allowed.
`'mdia'` (page 47)
The media for the track. Only one allowed.

Optional Child Atoms

`'clip'` (page 11)
Defines the clipping region for the track. Only one allowed.
`'matt'` (page 45)
Defines a matte for a track's media. Only one allowed.
`'edts'` (page 27)
Defines the portions of the track's media that are going into the track. Only one allowed.
`'tref'` (page 80)
`Track` reference atom. Only one allowed.
`'load'` (page 44)
Contains preloading information for a track. Only one allowed.
`'imap'` (page 39)
An input map. Only one allowed.
`'udta'` (page 84)
User data atom. Only one allowed.

Discussion

A movie may consist of one or more tracks. Each track is independent of the other tracks in the movie and carries its own temporal and spatial information. Each track atom contains an associated media atom. Note that there must be at least one media track within a QuickTime movie. All media tracks that are present must remain in the movie, even if the media data within them is not referenced by the hint tracks. After deleting all hint tracks, the entire unhinted movie must remain.

Programming Info

C interface file: `MoviesFormat.h`

See Also

See the `TrackDirectoryEntry` structure. For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'tref' `Track` reference atom.

This is a QT container atom; it is not declared in the header files. You can create it with `QTNewAtomContainer` and insert it with `QTInsertChild`, using the following parameters:

Fields

`atomType`

Discussion

Constant `TrackHeaderAID`, designating atom type `'tref'`.

`data`

Discussion

One track reference atom of a type listed below.

Parent Atom

`'trak'` (page 79)
Parent atom can contain only one atom of this type.

Required Child Atoms (one from this list)

`'tmcd'` (page 78)
Time code track reference type atom. Only one allowed.
`'chap'` (page 11)
Chapter or scene list track reference type atom. Only one allowed.
`'sync'` (page 75)
Synchronization track reference type atom. Only one allowed.
`'scpt'` (page 63)
Transcript track reference type atom. Only one allowed.
`'ssrc'` (page 67)
Nonprimary source track reference type atom. Only one allowed.
`'hint'` (page 34)
Hint track reference type atom. Only one allowed.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'trpy' Total number of bytes that will be sent, including 12-byte RTP headers, but not including network headers.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields
`atomType`

Discussion
Value is `'trpy'`.

`data`

Discussion
8 bytes.

Parent Atom

`'hinf'` (page 34)
Parent atom can contain only one atom of this type.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'twdt' Tween data type atom.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields
`atomType`

Discussion
Value is `'twdt'`.

`data`

Discussion
Tween data.

Parent Atom

`'twen'` (page 82)

Parent atom can contain only one atom of this type.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'twdu' Tween duration atom.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields

`atomType`

Discussion

Constant `kTweenDuration`, designating atom type `'twdu'`.

`data`

Discussion

Tween duration data.

Parent Atom

`'twen'` (page 82)

Parent atom can contain only one atom of this type.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'twen' Tween entry atom.

This is a QT container atom; it is not declared in the header files. You can create it with `QTNewAtomContainer` and insert it with `QTInsertChild`, using the following parameter:

Fields

`atomType`

Discussion

Constant `kTweenEntry`, designating atom type `'twen'`.

Required Child Atoms

`'twst'` (page 83)

Tween start atom. Only one allowed.

`'twdu'` (page 82)

Tween duration atom. Only one allowed.

`'twdt'` (page 81)

Tween data type atom. Only one allowed.

`'twnt'` (page 82)

Tween type atom. Only one allowed.

See Also

See the `TweenSequenceEntryRecord` structure. For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'twnt' Tween type atom.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields
`atomType`

Discussion
Constant `kTweenType`, designating atom type `'twnt'`.

`data`

Discussion
A tween type; see `Tween Types`.

Parent Atom

`'twen'` (page 82)
Parent atom can contain only one atom of this type.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'twst' Tween start offset atom.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields
`atomType`

Discussion
Constant `kTweenStartOffset`, designating atom type `'twst'`.

`data`

Discussion
Tween start offset.

Parent Atom

`'twen'` (page 82)
Parent atom can contain only one atom of this type.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

' ty' Input atom type.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields
`atomType`

Discussion
Value is `'  ty'`; first and second characters are spaces.

`data`

Discussion
`Track` input atom type code.

Parent Atom

`' in'` (page 43)
Parent atom can contain only one atom of this type.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'udta' Holds one or more structures of movie user data.

```
struct UserDataAtom {
    long              size;
    long              atomType;
    MoviesUserData    userData[1];
};
```

Fields

`size`

Discussion
The size in bytes of this atom structure.

`atomType`

Discussion
Constant `UserDataAID`, designating atom type `'udta'`.

`userData`

Discussion
An array of `MoviesUserData` data structures, which contain the actual data for this atom. The currently defined types are listed below.

Parent atom (one or the other)

`'moov'` (page 52)
Parent atom can contain any number of atoms of this type.
`'trak'` (page 79)
Parent atom can contain any number of atoms of this type.

Optional child atom

`' ccpy'` (page 13)
Copyright statement.
`' cday'` (page 14)
Date the movie content was created.
`' cdir'` (page 15)
Name of movie's director.
`' ced1'` (page 15)
First edit date and description. Similar for ' ced2' through ' ced9'.
`' cfmt'` (page 16)
Indication of movie format (computer-generated, digitized, and so on).
`' cinf'` (page 16)
Information about the movie.
`' cprd'` (page 17)
Name of movie's producer.
`' cprf'` (page 17)
Names of performers.
`' creq'` (page 18)
Special hardware and software requirements.
`' csrc'` (page 19)
Credits for those who provided movie source content.

`' cwrt'` (page 19)
Name of movie's writer.
`'WLOC'` (page 91)
Default window location for movie. Two 16-bit values, `{x,y}`.
`'name'[userdata]` (page 54)
Name of object.
`'LOOP'` (page 45)
Looping. Long integer indicating looping `style`. 0 for none, 1 for looping, 2 for palindromic looping.
`'SelO'` (page 64)
Play selection only. `Byte` indicating that only the selected area of the movie should be played.
`'AllF'` (page 10)
Play all frames. `Byte` indicating that all frames of video should be played, regardless of timing.

Discussion
Inside the user data atom is a list of atoms describing each piece of user data. Each data element contains size and type information along with the data. Furthermore, for historical reasons, the list of atoms is optionally terminated by a 32-bit integer set to 0. If you are writing a program to read user data atoms, you should allow for the terminating 0. However, if you are writing a program to create user data atoms, you can safely leave out the trailing 0.

Programming Info
C interface file: `MoviesFormat.h`

See Also

See the `MoviesUserData` structure. For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'url ' Contains a URL for a sprite.

This is a QT container atom; it is not declared in the header files. You can create it with `QTNewAtomContainer` and insert it with `QTInsertChild`, using the following parameter:

Fields
`atomType`

Discussion
Value is `'url '`.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'uses' Lets a sprite specify the subset of images that its image index property can refer to.

This is a QT container atom; it is not declared in the header files. You can create it with `QTNewAtomContainer` and insert it with `QTInsertChild`, using the following parameter:

Fields
`atomType`

Discussion
Value is `'uses'`.

Parent Atom
`'sprt'` (page 66)
A key frame sprite definition.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'vars' Contains variables for a sprite.

This is a QT container atom; it is not declared in the header files. You can create it with `QTNewAtomContainer` and insert it with `QTInsertChild`, using the following parameter:

Fields
`atomType`

Discussion
Value is `'vars'`.

Optional Child Atoms

`'flov'` (page 31)
Contains a floating-point variable for a sprite. Multiple atoms allowed.
`'strv'` (page 70)
Contains a string variable for a sprite. Multiple atoms allowed.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'vide' Contains compression information for the Base Image Exporter.

This is a QT atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameter:

Fields
`atomType`

Discussion
Constant `VideoMediaType`, designating atom type `'vide'`; see `Component Identifiers`.

Optional Child Atoms

`'dasz'` (page 23)
Only one allowed. If present, it specifies a desired data size. The base exporter repeats compression attempts, decreasing the `quality` parameter until it reaches this size or lower, or it runs out of patience.
`'reso'` (page 59)
Only one allowed. If present, it specifies the resolution for the `pixmap` passed to the compressor.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'vmhd'[media] Stores handler-specific information for video media in a track.

```
struct VideoMediaInfo {
    long                        size;
    long                        atomType;
    VideoMediaInfoHeaderAtom    header;
    HandlerAtom                 dataHandler;
    DataInfoAtom                dataInfo;
    SampleTableAtom             sampleTable;
};
```

Fields
`size`

Discussion
The size in bytes of this atom structure.

`atomType`

Discussion
Constant `VideoMediaInfoAID`, designating atom type `'vmhd'`.

`header`

Discussion
A `'vmhd'[transfer]` (page 87) atom, which defines the graphics transfer mode for this video media.

`dataHandler`

Discussion
A `'hdlr'` (page 33) atom that specifies the component that is to handle this media.

`dataInfo`

Discussion
A `'dinf'` (page 25) atom, which specifies where the video media data is stored.

`sampleTable`

Discussion
A `'stbl'` (page 68) atom, which tells the media handler how to locate and interpret data samples.

Discussion
This atom stores handler-specific information for the media that constitutes a video track. A video media handler uses this information to map from media time to media data. Another type of media handler would not know how to interpret this information.

Programming Info
C interface file: `MoviesFormat.h`

See Also
See the `'minf'[sound]` (page 50) atom for sound media and the `'minf'[generic]` (page 49) atom for media other than video or sound. Also see the `VideoMediaInfoHeader` structure. For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'vmhd'[transfer] Defines the graphics transfer characteristics for a video media.

```
struct VideoMediaInfoHeaderAtom {
    long                    size;
    long                    atomType;
    VideoMediaInfoHeader    vmiHeader;
};
```

Fields
`size`

Discussion
The size in bytes of this atom structure.

`atomType`

Discussion
Constant `VideoMediaInfoHeaderAID`, designating atom type `'vmhd'`.

`vmiHeader`

Discussion
A `VideoMediaInfoHeader` data structure, which contains the actual data for this atom.

Programming Info
C interface file: `MoviesFormat.h`

See Also

For the structure that contains this atom, see `'minf'[video]` (page 51). Also see the `VideoMediaInfoHeader` structure. For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'vrcp' Custom cursor atom parent.

This is a QT container atom; it is not declared in the header files. You can create it with `QTNewAtomContainer` and insert it with `QTInsertChild`, using the following parameter:

Fields
`atomType`

Discussion
Constant `kQTVRCursorParentAtomType`, designating atom type `'vrcp'`.

Required Child Atoms

`'CURS'` (page 22)
Custom cursor child atom. Multiple atoms of this type allowed.
`'crsr'` (page 20)
Color custom cursor child atom. Multiple atoms of this type allowed.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'vrni' QTVR node ID atom.

This is a QT container atom; it is not declared in the header files. You can create it with `QTNewAtomContainer` and insert it with `QTInsertChild`, using the following parameter:

Fields
`atomType`

Discussion
Constant `kQTVRNodeIDAtomType`, designating atom type `'vrni'`.

Parent Atom

`'vrnp'` (page 88)
Parent atom can contain any number of atoms of this type.

Required Child Atoms

`'nloc'` (page 55)
QTVR node location atom. Only one allowed.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'vrnp' QTVR node parent atom.

This is a QT container atom; it is not declared in the header files. You can create it with `QTNewAtomContainer` and insert it with `QTInsertChild`, using the following parameter:

Fields
`atomType`

Discussion
Constant `kQTVRNodeParentAtomType`, designating atom type `'vrnp'`.

Required Child Atoms

`'vrni'` (page 88)
QTVR node ID atom. Any number allowed.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'vrsc' VR world header atom.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields
`atomType`

Discussion
Constant `kQTVRWorldHeaderAtomType`, designating atom type `'vrsc'`.

`data`

Discussion
Contains the name of the scene and the default node ID.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'vrsg' Contains the name of a VR scene.

This is a QT leaf atom that contains a struct in its data field. Using the constant `kQTVRStringAtomType` and a pointer to the atom, you can access its data with `QTGetAtomDataPtr` and change it with `QTSetAtomData`. The struct is declared as follows:

```
struct QTVRStringAtom {
    UInt16          stringUsage;
    UInt16          stringLength;
    unsigned char   theString[4];
};
```

Fields
`stringUsage`

Discussion
Unused field.

`stringLength`

Discussion
The length of the name string in bytes.

`theString`

Discussion
The name as a C string.

Discussion
One leaf atom of this type is contained in a VR world container. You can get a pointer to this container by calling `QTVRGetVRWorld`. One of this atom's siblings in the VR world is a `'vrsc'` (page 89) atom, which contains the atom ID of this atom in its `nameAtomID` field. The following code illustrates a function that returns the name of a VR node as a Pascal string, given the node's ID:

```
OSErr MyGetNodeName (QTVRInstance theInstance, UInt32 theNodeID,
```

```
                    StringPtr theStringPtr)
{
    OSErr                       theErr =noErr;
    QTAtomContainer             theNodeInfo;
    QTVRNodeHeaderAtomPtr       theNodeHeader;
    QTAtom                      theNodeHeaderAtom =0;

    // Get the node information atom container
    theErr =QTVRGetNodeInfo(theInstance, theNodeID, &theNodeInfo);

    // Get the node header atom.
    if (!theErr)
        theNodeHeaderAtom =QTFindChildByID(theNodeInfo,
                                           kParentAtomIsContainer,
                                            kQTVRNodeHeaderAtomType, 1, nil);
    if (theNodeHeaderAtom !=0) {
        QTLockContainer(theNodeInfo);

        // Get a pointer to the node header atom data.
        theErr =QTGetAtomDataPtr(theNodeInfo, theNodeHeaderAtom, nil,
                                 (Ptr *)&theNodeHeader);
        // See if there is a name atom.
        if (!theErr && theNodeHeader->
nameAtomID !=0) {
            QTAtom theNameAtom;
            theNameAtom =QTFindChildByID(theNodeInfo,
                         kParentAtomIsContainer, kQTVRStringAtomType,
                         theNodeHeader->
nameAtomID, nil);
            if (theNameAtom !=0) {
                VRStringAtomPtr theStringAtomPtr;

                // Get a pointer to the name atom data; copy it into string
                theErr =QTGetAtomDataPtr(theNodeInfo, theNameAtom, nil,
                                         (Ptr *)&theStringAtomPtr);
                if (!theErr) {
                    short theLen =theStringAtomPtr->
stringLength;
                    if (theLen >
255)
                        theLen =255;
                    BlockMove(theStringAtomPtr->
theString,
                              &theStringPtr[1], theLen);
                    theStringPtr[0] =theLen;
                }
            }
        }
        QTUnlockContainer(theNodeInfo);
    }
    QTDisposeAtomContainer(theNodeInfo);
    return(theErr);
}
```

Programming Info

C interface file: `QuickTimeVRFormat.h`

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'wide' Wide atom name placeholder atom.

This is a QT leaf atom; it is not declared in the header files. You can create it with `QTInsertChild` using the following parameters:

Fields
`atomType`

Discussion
Constant `WideAtomPlaceholderType`, designating atom type `'wide'`.

`data`

Discussion
8 bytes of placeholder space to allow an atom to be converted from a 32-bit to a 64-bit atom.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'WLOC' User data list entry atom: default window location for movie.

```
struct MoviesUserData {
    long    size;
    long    udType;
    char    data[1];
};
```

Fields
`size`

Discussion
The size in bytes of this atom structure.

`udType`

Discussion
Value is `'WLOC'`.

`data`

Discussion
2 16-bit values, `{x,y}`.

Parent Atom

`'udta'` (page 84)
Parent atom can contain only one atom of this type.

Programming Info
C interface file: `MoviesFormat.h`

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

'wtxt' Parent atom for hypertext items.

This is a QT container atom; it is not declared in the header files. You can create it with `QTNewAtomContainer` and insert it with `QTInsertChild`, using the following parameter:

Fields
`atomType`

Discussion
Value is `'wtxt'`.

Required Child Atoms

`'strt'` (page 70)
Defines the starting offset of hypertext in a text stream. Only one allowed.
`'end '` (page 29)
Defines the ending offset of hypertext in a text stream. Only one allowed.

Optional Child Atoms

`'htxt'` (page 37)
Multiple atoms allowed.

See Also

For general information about atoms, see *Inside QuickTime: QuickTime File Format*.

QuickTime Public Resources

Resources

`'atms'`

Lists effect and parameter description atoms for effect components.

```
type 'atms' {
    longint;                                        // root atom count
    array AtomArray {
        literal    longint;                         // atomType
        longint;                                    // atomID
        longint    noChildren =0;                  // children
        longint =$$CountOf(AtomData);
        array AtomData {
            switch {
                case long:
                    key literal longint ='long';
                    pstring;                        // data
                case short:
                    key literal longint ='shrt';
                    pstring;                        // data
                case noMininumFixed:
                    key literal longint ='nmiF';
                    pstring = "";                   // data
                case noMaximumFixed:
                    key literal longint ='nmaF';
                    pstring = "";                   // data
                case noMininumDouble:
                    key literal longint ='nmiD';
                    pstring = "";                   // data
                case noMaximumDouble:
                    key literal longint ='nmaD';
                    pstring = "";                   // data
                case fixed:
                    key literal longint ='fixd';
                    pstring;                        // data
                case double:
                    key literal longint ='doub';
                    pstring;                        // data
                case string:
                    key literal longint ='str ';
                    pstring;                        // data
                case lstring:
                    key literal longint ='lstr';
                LongStringStart:
                    longint =
                    ((LongStringEnd[$$ArrayIndex(AtomArray),
                        $$ArrayIndex(AtomData)] -
                        LongStringStart[$$ArrayIndex(AtomArray),
                        $$ArrayIndex(AtomData)]) >
>
3) - 4;
                    hex string
                        [$$Word(LongStringStart[$$ArrayIndex(AtomArray),
                        $$ArrayIndex(AtomData)]) - 4];
                LongStringEnd:
                case OSType:
                    key literal longint ='osty';
                    pstring;                        // data
            };
        };
    };
};
```

Discussion
The `'atms'` resource for a video effect contains two sets of information. The first set contains the effect information that is used to construct the standard parameters dialog box. This includes items such as the name of the effect and optional copyright information. The second set contains a description of each parameter that the effect takes. If the effect does not take parameters, there is no information in this set.

Version Notes
Introduced in QuickTime 6.

Programming Info
Resource accessibility: Can be made public
Rez source file: `ImageCodec.r`
Programming summary: `Component Public Resources`

'avvc'

Lists AVI four cc types for compressor components.

```
type 'avvc' {
    array {
        literal    longint;    // avi four cc type
    };
};
```

Version Notes
Introduced in QuickTime 6.

Programming Info
Resource accessibility: Can be made public
Rez source file: `ImageCodec.r`
Programming summary: `Component Public Resources`

'avvd'

Lists AVI four cc types for decompressor components.

```
type 'avvd' {
    array {
        literal    longint;    // avi four cc type
    };
};
```

Version Notes
Introduced in QuickTime 6.

Programming Info
Resource accessibility: Can be made public
Rez source file: `ImageCodec.r`
Programming summary: `Component Public Resources`

'cdci'

Contains codec characteristics.

```
type 'cdci' {
    pstring[31];
    hex integer    version;
    hex integer    revlevel;
    hex longint    vendor;
    hex longint    decompressFlags;
    hex longint    compressFlags;
    hex longint    formatFlags;
    byte           compressionAccuracy;
    byte           decompressionAccuracy;
    integer        compressionSpeed;
    integer        decompressionSpeed;
    byte           compressionLevel;
    byte           resvd;
    integer        minimumHeight;
    integer        minimumWidth;
    integer        decompressPipelineLatency;
    integer        compressPipelineLatency;
    longint        privateData;
};
```

Version Notes
Introduced in QuickTime 6.

Programming Info
Resource accessibility: Private
Rez source file: `ImageCodec.r`
Programming summary: `Component Public Resources`

'cdec'

Contains a codec string.

```
type 'cdec' {
    hex    string;
};
```

Version Notes
Introduced in QuickTime 6.

Programming Info
Resource accessibility: Private
Rez source file: `ImageCodec.r`
Programming summary: `Component Public Resources`

'cpix'

Lists supported pixel formats for a codec compressor.

```
type 'cpix' {
    array {
        literal    longint;
    };
};
```

Discussion
A `'cpix'` resource is an array of 4-character codes (such as '2vuy' or `'yuvs'`) that define the pixel formats a codec can accept. You can use this array to list any pixel formats that your codec prefers to straight RGB. An application can get the list, by calling `GetComponentPublicResource`, to see what kind of graphics world it should construct.

Version Notes
Introduced in QuickTime 6.

Programming Info
Resource accessibility: Can be made public
Rez source file: `ImageCodec.r`
Programming summary: `Component Public Resources`

`'dlle'`

Contains a string for a multiplatform component.

```
type 'dlle' {
    cstring;
};
```

Version Notes
Introduced in QuickTime 6.

Programming Info
Resource accessibility: Private
Rez source file: `Components.r`
Programming summary: `Component Public Resources`

`'mcfg'`

Lists characteristics of files supported by a graphics importer component.

```
type 'mcfg' {
    longint =kQTMediaConfigResourceVersion;   // resource version (long)
    // version of the component this applies to
    longint kVersionDoesntMatter =0;
    // array, one entry for each media type
    longint =$$Countof(MIMEInfoArray);
    array MIMEInfoArray {
        literal longint;  // ID of the group this type belongs with:
                          // OSType, a kQTMediaConfigStreamGroupID, etc.
        literal longint;  // MIME config flags:
                          // unsigned long, a kQTMediaConfigCanUseApp, etc.
        literal longint;  // MacOS file type when saved (OSType)
        literal longint;  // MacOS file creator when saved (OSType)
        literal longint;          // component type (OSType)
        literal longint;          // component subtype (OSType)
        literal longint;          // component manufacturer (OSType)
        unsigned hex longint;     // component flags
        // component flags mask
        unsigned hex longint kAnyComponentFlagsMask =0;
        literal longint;          // default file extension (OSType)
                                  // all caps to match subType
                                  // of eat and grip components
        literal longint;          // QT file group:
                                  // OSType, a kQTMediaInfoNetGroup, etc.
        longint =$$Countof(QTMediaSynonymsArray);
        array QTMediaSynonymsArray {
            pstring;              // array of media type synonyms
        };
        align long;               // align
        wide array [5] {
            pstring;
            // array of 5 Pascal strings:
            //     + media type description
            //     + file extension(s)
            //     + opening application name
            //     + missing software description
            //     + vendor info string (copyright, version, etc)
        };
        align long;               // align
        // array of MIME types that describe this
        // (eg. audio/mpeg, audio/x-mpeg, etc.)
        longint =$$Countof(MIMETypeArray);
        array MIMETypeArray {
            pstring;
        };
        align long;               // align
    };
};
```

Version Notes
Introduced in QuickTime 6.

Programming Info
Resource accessibility: Private
Rez source file: `QuicktimeComponents.r`
Programming summary: `Component Public Resources`

'mgrp'

Lists MIME groups supported by a graphics importer component.

```
type 'mgrp' {
    longint =kQTMediaGroupResourceVersion;    // resource version (long)
    // component version this applies to
    longint kVersionDoesntMatter =0;
    // array of group information
    // (optional unless you are defining new group(s))
    longint =$$Countof(MIMEGroupArray);
    array MIMEGroupArray {
        literal longint;                // group ID (OSType)
        pstring;                        // name of the grouping
        pstring;                        // description
        align long;                     // align
    };
};
```

Version Notes
Introduced in QuickTime 6.

Programming Info
Resource accessibility: Private
Rez source file: `QuicktimeComponents.r`
Programming summary: `Component Public Resources`

'mime'[resource]

Lists MIME types supported by a movie importer or exporter component.

```
type 'mime' {
    longint =0;    // 10 bytes of reserved
    longint =0;
    integer =0;
    integer =0;    // 2 bytes of lock count
    parentStart:
        longint =( (parentEnd - parentStart) / 8 );   // size of this atom
        literal longint ='sean';                      // atom type
        longint =1;                                   // atom ID
        integer =0;
        integer = $$CountOf(AtomArray);
        longint =0;
        array AtomArray {
            atomStart:
                // size of this atom
                longint =((atomEnd[$$ArrayIndex(AtomArray)] -
                    atomStart[$$ArrayIndex(AtomArray)]) / 8);
                literal longint;                       // atom type
                longint;                               // atom ID
                integer =0;
                integer =0;                           // no children
                longint =0;
                string;
            atomEnd:
        };
    parentEnd:
};
```

Discussion
Every import component should include a `'thnr'` (page 112) resource holding the same data that `MovieImportGetMIMETypeList` would return. By including this public resource, QuickTime and applications don't need to open the import component and call `MovieImportGetMIMETypeList` to determine the MIME types the importer supports. In the absence of this resource, QuickTime and applications will use `MovieImportGetMIMETypeList`. This resource's public type and ID should be `'mime'` and 1. Here is an example of such a list:

```
resource 'thnr' (kMyImportComponentResID)  {
    'mime', 1, 0,
    'mime', kMyImportMIMETypeListResID, 0
}
```

Version Notes
Introduced in QuickTime 6.

Programming Info
Resource accessibility: Can be made public
Rez source file: `QuicktimeComponents.r`
Programming summary: `Component Public Resources`

`'pcki'`

Lists streaming payload media supported by a packetizer component.

```
type 'pcki' {
    array infoArray {
        align long;
        hex longint    mediaType;
        hex longint    dataFormat;
        hex longint    vendor;
        hex longint    capabilityFlags;
        byte           canPackMatrixType;
        byte =0;
        byte =0;
        byte =0;
        longint =$$CountOf(characteristicArray);    // array size
        array characteristicArray {
            hex longint    tag;
            hex longint    value;
        };
        hex longint    payloadFlags;
        byte           payloadID;                    // if static payload
        byte =0;
        byte =0;
        byte =0;
        cstring;
    };
};
```

Discussion
Every packetizer must provide a public resource of type `'pcki'`, which contains information about its capabilities. This information lists the `media` types and compression formats the packetizer can work with. It also lists the track characteristics the packetizer can work with, such as layers or transformation matrices. In addition, it provides information about the packetizer's performance characteristics, such as its speed or ability to recover from packet loss. QuickTime selects the packetizer best suited to a stream's current media, compression format, and track characteristics. If there are multiple packetizers that can work with a given track, QuickTime picks the one with the best performance.

Version Notes
Introduced in QuickTime 6.

Programming Info
Resource accessibility: Can be made public
Rez source file: `QTStreamingComponents.r`
Programming summary: `Component Public Resources`

`'qter'`

Stores error messages for `QTAddMovieError`.

```
type 'qter' {
    longint =$$Countof(ErrorSpec);
    wide array ErrorSpec {
    longint; // error code used to find this error
    longint // error type
        kQuickTimeErrorNotice =1,
        kQuickTimeErrorWarning =2,
        kQuickTimeErrorError =3;
    // In the following strings, ^FILENAME, ^APPNAME, ^0, ^1, etc will be
    // replaced as appropriate.
    pstring; // main error string
    pstring; // explanation error string
    pstring; // technical string (not displayed
             // to user except in debug cases)
    align long;
    };
};
```

Version Notes
Introduced in QuickTime 6.

Programming Info
Resource accessibility: Can be made public
Rez source file: `Movies.r`

`'rsmi'`

Lists the characteristics of streaming payloads supported by a reassembler component.

```
type 'rsmi' {
    array infoArray {
        align long;
        longint =$$CountOf(characteristicArray);    // array size
        array characteristicArray {
            hex longint    tag;
            hex longint    value;
        };
        hex longint    payloadFlags;
        byte           payloadID;    // if static payload
        byte =0;
        byte =0;
        byte =0;
        cstring;
    };
};
```

Discussion
Every reassembler must provide a public resource of type `'rsmi'`, which contains information about its capabilities. This information lists the RTP payload types the reassembler can work with, as well as the reassembler's speed and ability to recover from lost packets. If more than one reassembler is available for a given RTP payload type, QuickTime chooses the one with the best performance characteristics, such as highest speed or best ability to deal with packet loss.

Version Notes
Introduced in QuickTime 6.

Programming Info
Resource accessibility: Can be made public
Rez source file: `QTStreamingComponents.r`
Programming summary: `Component Public Resources`

'skcr'

Defines a media skin content region.

```
// no declaration
```

Discussion
The content of this resource is currently a 1-bit `'pict'` image.

Version Notes
Introduced in QuickTime 6.

Programming Info
Resource accessibility: Can be made public
Programming summary: `Component Public Resources`

'skgr'

Defines a media skin drag region.

```
// no declaration
```

Discussion
The content of this resource is currently a 1-bit `'pict'` image.

Version Notes
Introduced in QuickTime 6.

Programming Info
Resource accessibility: Can be made public
Programming summary: `Component Public Resources`

'snd '

Lists sound commands supported by a sound component.

```
type 'snd ' {
    switch {
        case FormatOne:
            key unsigned integer =$0001;
            unsigned integer =$$CountOf(Synthesizers);
            wide array Synthesizers {
                // Resource ID of synthesizer/modifer
                integer    squareWaveSynth    =$0001,
                           waveTableSynth     =$0003,
                           sampledSynth       =$0005;
                longint;                             // init parameter
                };
        case FormatTwo:
            key unsigned integer =$0002;
            integer free =0, keepInMemory =256+1; // Space for refe count
        };
        unsigned integer =$$CountOf(SoundCmnds);
        wide array SoundCmnds {
            boolean     noData, hasData;
            switch {
                case nullCmd:
                    key bitstring[15] =0;
                    fill word;                       // Param 1 =nil
                    fill long;                       // Param 2 =nil
                case quietCmd:
                    key bitstring[15] =3;
                    fill word;                       // Param 1 =nil
                    fill long;                       // Param 2 =nil
                case flushCmd:
                    key bitstring[15] =4;
                    fill word;                       // Param 1 =nil
                    fill long;                       // Param 2 =nil
                case waitCmd:
                    key bitstring[15] =10;
                    integer    oneSecond =2000;    // Duration
                    fill long;                       // Param 2 =nil
                case pauseCmd:
                    key bitstring[15] =11;
                    fill word;                       // Param 1 =nil
                    fill long;                       // Param 2 =nil
                case resumeCmd:
                    key bitstring[15] =12;
                    fill word;                       // Param 1 =nil
                    fill long;                       // Param 2 =nil
                case callBackCmd:
                    key bitstring[15] =13;
                    integer;                         // User-defined
                    longint;                         // User-defined
                case syncCmd:
                    key bitstring[15] =14;
                    integer;                         // Count
                    longint;                         // Identifier
                case emptyCmd:
                    key bitstring[15] =15;
                    fill word;                       // Param 1 =nil
                    fill long;                       // Param 2 =nil
                case freqDurationCmd:
                    key bitstring[15] =40;
```

```
                integer    oneSecond =2000;    // Duration
                longint;                       // Frequency
            case restCmd:
                key bitstring[15] =41;
                integer    oneSecond =2000;    // Duration
                fill long;                     // Param 2 =nil
            case freqCmd:
                key bitstring[15] =42;
                fill word;                     // Param 1 =nil
                longint;                       // Frequency
            case ampCmd:
                key bitstring[15] =43;
                integer;                       // Amplitude
                fill long;                     // Param 2
            case timbreCmd:
                key bitstring[15] =44;
                integer sineWave, squareWave =255;     // Timbre
                fill long;                     // Param 2
            case waveTableCmd:
                key bitstring[15] =60;
                unsigned integer;              // Length
                longint;                       // Pointer to table
            case phaseCmd:
                key bitstring[15] =61;
                integer;                       // Shift
                longint;                       // chanPtr
            case soundCmd:
                key bitstring[15] =80;
                fill word;                     // Param 1 =nil
                longint;                       // Pointer to sound
            case bufferCmd:
                key bitstring[15] =81;
                fill word;                     // Param 1 =nil
                longint;                       // Pointer to buffer
            case rateCmd:
                key bitstring[15] =82;
                fill word;                     // Param 1 =nil
                longint;                       // Rate
        };
    };
    array DataTables {
        DataTable:
            fill long;                         // Pointer to data
        SampleCnt:
            unsigned longint;                  // # of sound samples
            unsigned hex longint Rate22K =$56EE8BA3;  // Sampling rate
            unsigned longint;                  // Start of loop
            unsigned longint;                  // End of loop
            hex byte;                          // encode (header type)
            hex byte;                          // baseFrequency
            hex string [$$Long(SampleCnt[$$ArrayIndex(DataTables)])];
    };
};
```

Version Notes
Introduced in QuickTime 6.

Programming Info
Resource accessibility: Private
Rez source file: `Sound.r`
Programming summary: `Component Public Resources`

'src#'

Lists a movie exporter component's supported media types and the minimum and maximum number of sources for each.

```
type 'src#' {
    longint =$$CountOf(SourceArray);
    longint =0;                     // reserved
    array SourceArray {
        literal longint;            // Media type of source
        // min number of sources of this kind required; 0 if none required
        integer;
        // max number of sources of this kind allowed;
        //   65535 if unlimited allowed
        integer;
        longint    isMediaType              =0x01,
                   isMediaCharacteristic    =0x02,
                   isSourceType             =0x04;
    };
};
```

Version Notes
Introduced in QuickTime 6.

Programming Info
Resource accessibility: Can be made public
Rez source file: `QuicktimeComponents.r`
Programming summary: `Component Public Resources`

'stg#'

Lists QuickTime's presets.

```
type 'stg#' {
    hex longint;                    // flags
    longint =$$CountOf(PresetDescriptionArray);
    longint =0;
    array PresetDescriptionArray {
        literal longint;            //  preset key ID
        unsigned hex longint noFlags =0,
                             kQTPresetInfoIsDivider =1;   //  preset flags
        literal longint;            //  preset resource type
        integer;                    //  preset resource ID
        integer =0;                //  padding but also reserved
        integer;                    //  preset name string list ID
        integer;                    //  preset name string index
        integer;                    //  preset description string list ID
        integer;                    //  preset description string index
    };
};
```

Version Notes
Introduced in QuickTime 6.

Programming Info
Resource accessibility: Private
Rez source file: `QuicktimeComponents.r`
Programming summary: `Component Public Resources`

'stgp'

Lists QuickTime's preset platforms.

```
type 'stgp' {
    longint =0;                   // reserved
    literal longint;               // default settings list resource type
    integer;                       // default settings list resource id
    integer =$$CountOf(SettingsPlatformInfo);
    wide array SettingsPlatformInfo {
        unsigned hex longint =0;  // reserved
        literal longint;           // platform settings list resource Type
        integer;                   // platform settings list resource ID
        // platform type (response from gestaltSysArchitecture)
        integer    platform68k =1,
                   platformPowerPC =2,
                   platformInterpreted =3,
                   platformWin32 =4;
    };
};
```

Version Notes
Introduced in QuickTime 6.

Programming Info
Resource accessibility: Private
Rez source file: `QuickTimeComponents.r`
Programming summary: `Component Public Resources`

'stri'

Contains a component information string.

```
type 'stri' {
     pstring;    // string
};
```

Version Notes
Introduced in QuickTime 6.

Programming Info
Resource accessibility: Private
Rez source file: `Components.r`
Programming summary: `Component Public Resources`

'strn'

Contains a component name string.

```
type 'strn' {
     pstring;    // string
};
```

Version Notes
Introduced in QuickTime 6.

Programming Info
Resource accessibility: Private
Rez source file: `Components.r`
Programming summary: `Component Public Resources`

'sttg'

Lists QuickTime's presets.

```
// no declaration
```

Version Notes
Introduced in QuickTime 6.

Programming Info
Resource accessibility: Private
Programming summary: `Component Public Resources`

'thga'

Lists the characteristics of a component resource alias.

```
type 'thga' {
    literal longint;                                    // type
    literal longint;                                    // subtype
    literal longint;                                    // manufacturer
    unsigned hex longint;                               // component flags
    unsigned hex longint kAnyComponentFlagsMask =0; // component flags mask
    literal longint;                                    // code type
    integer;                                            // code ID
    literal longint;                                    // name type
    integer;                                            // name ID
    literal longint;                                    // info type
    integer;                                            // info ID
    literal longint;                                    // icon type
    integer;                                            // icon ID
    literal longint;                                    // type
    literal longint;                                    // subtype
    literal longint;                                    // manufacturer
    unsigned hex longint;                               // component flags
    unsigned hex longint kAnyComponentFlagsMask =0; // component flags mask
    #if thng_RezTemplateVersion >
=2
        literal longint;                                // resource map type
        integer;                                        // resource map id
        integer    cmpAliasNoFlags =0,
                   cmpAliasOnlyThisFile =1;         // alias flags
    #endif
};
```

Version Notes
Introduced in QuickTime 6.

Programming Info
Resource accessibility: Private
Rez source file: `Components.r`
Programming summary: `Component Public Resources`

'thn#'

Lists a component's load order dependencies.

```
type 'thn#' {
    array {
        literal    longint;    // code type
        integer;               // code ID
    };
};
```

Version Notes
Introduced in QuickTime 6.

Programming Info
Resource accessibility: Private
Rez source file: `Components.r`
Programming summary: `Component Public Resources`

'thnd'

Lists a component's dependencies.

```
type 'thnd' {
    longint =$$CountOf(ComponentDependency);
    wide array ComponentDependency {
        literal longint;         // type
        literal longint;         // subtype
        literal longint;         // manufacturer
        unsigned hex longint;    // component flags
        unsigned hex longint    kAnyComponentFlagsMask =0; // flags mask
    };
};
```

Version Notes
Introduced in QuickTime 6.

Programming Info
Resource accessibility: Private
Rez source file: `Components.r`
Programming summary: `Component Public Resources`

'thng'

Lists the characteristics of a component resource.

```
type 'thng' {
    literal      longint;                              // type
    literal      longint;                              // subtype
    literal      longint;                              // manufacturer
    unsigned hex longint;                              // component flags
    unsigned hex longint kAnyComponentFlagsMask =0; // component flags mask
    literal      longint;                              // code type
    integer;                                           // code ID
    literal      longint;                              // name type
    integer;                                           // name ID
    literal      longint;                              // info type
    integer;                                           // info ID
    literal      longint;                              // icon type
    integer;                                           // icon ID
    #if thng_RezTemplateVersion >
=1
        unsigned hex longint;                          // version
        longint;                                       // registration flags
        integer;                                  // resource ID of icon family
        longint =$$CountOf(ComponentPlatformInfo);
        wide array ComponentPlatformInfo {
            unsigned hex    longint;                   // component flags
            literal         longint;                   // code type
            integer;                                   // code ID
            integer    platform68k =1,                // platform type
                       platformPowerPC =2,
                       platformInterpreted =3,
                       platformWin32 =4,
                       platformPowerPCNativeEntryPoint =5;
        };
        #if thng_RezTemplateVersion >
=2
            literal longint;                           // resource map type
            integer;                                   // resource map ID
        #endif
    #endif
};
```

Fields

`platform type`

Discussion

The response from `gestaltComponentPlatform` if available, or else from `gestaltSysArchitecture`.

`resource map`

Discussion

See the `'thnr'` (page 112) resource type.

Discussion

To associate a Public Resource Map with a component, the component's `'thng'` resource must be extended to include a references to a `'thnr'` (page 112) resource. This can be done when the value of `thng_RezTemplateVersion` is 2, by adding the `resource` type `'thnr'` and an ID. Here is an example:

```
resource 'thng' (512) {
    // component type, subtype, manufacturer, etc. go here
    'thnr', 512
};
```

Version Notes
Introduced in QuickTime 6.

Programming Info
Resource accessibility: Private
Rez source file: `Components.r`
Programming summary: `Component Public Resources`

`'thnr'`

Contains a public resource map for a component.

```
type 'thnr' {
    array {
        literal longint;        // resource type
        integer;                // resource id
        integer;                // unused flags
        literal    longint;     // Mac OS resource type
        integer;                // Mac OS resource ID
        integer    cmpResourceNoFlags =0,
                   cmpResourceCallComponent =1;    // flags
    };
};
```

Fields
`flags`

Discussion
Some components may need to build the contents of their public resources at run time. For example, the ColorSync visual effect's parameter list varies depending on what color matching methods are installed. In this case, its public component resource cannot be stored in its resource file, but instead must be dynamically created at run time. To indicate that a public resource cannot be loaded directly from a component's file, the component's `'thnr'` resource contains 0 in the `ID` field for that resource and the `cmpResourceCallComponent` flag is set to 1.

Discussion
Public resources are identified by a four-character `OSType` codes and ID numbers. Unlike private resources, however, a public resource's `OSType` code and ID do not have to be the same as the Mac OS resource type and ID that the resource is stored in. Consequently, a component that provides public resources must add a Public Resource Map to the component's `'thng'` (page 110) resource, giving the mapping between each public resource type and ID and the corresponding private resource type and ID. Here's an example of a Public Resource Map. It makes available two public resource, `'PICT'` 1 and `'PICT'` 2, which are stored in the component as Mac OS resources `'pict'` 128 and `'pict'` 129.

```
resource 'thnr' (512) {
    {
        'PICT', 1, 0, 'pict', 128, 0,
        'PICT', 2, 0, 'pict', 129, 0,
    }
}
```

Version Notes
Introduced in QuickTime 6.

Programming Info
Resource accessibility: Private
Rez source file: `Components.r`

Programming summary: `Component Public Resources`

See Also

For functions that access public resources, see `GetComponentPublicResource` and `GetComponentPublicResourceList`.

Document Revision History

This table describes the changes to *QuickTime Atoms and Resources Reference*.

Date	Notes
2006-05-23	New document, based on previously published material, that provides API details of QuickTime atoms and public resources.

Document Revision History