

Movie Resource Formats

This chapter describes the format of QuickTime movie resources. **Movie resources** are the data structures that provide the medium of exchange for movie data. Movie resources may be exchanged between applications on a single Macintosh computer or between applications on several Macintosh and non-Macintosh computers.

IMPORTANT

The information in this chapter is intended for developers who need to know about the content of movie resources. You need to learn about movie resources if you want to create movies on other computer environments and import them to the Macintosh environment, or if you want to interpret QuickTime movies on other types of computers. Developers of Macintosh applications do not need to know about the layout of movie resources—the Movie Toolbox functions handle the details of movie storage and exchange. ▲

This chapter describes **atoms**, the basic storage elements that, taken together, make up a movie resource.

The chapter is divided into the following major sections:

- “Storing Movies in Files” describes the two ways that QuickTime movies may be stored in files.
- “Atoms” describes the format and content of the most basic movie storage element and the standard atoms that may be found in a movie resource.
- “Overview of the Movie Resource Atom” provides a look at the movie resource structure of a QuickTime movie.
- “Using Media Information Atoms” provides examples of the media information atoms.

To understand fully the information presented in this chapter, you should be familiar with the Movie Toolbox (see the chapter “Movie Toolbox” in this book). In particular, you should read about the characteristics of movie, track, and media structures.

If you are developing a movie application that runs on another type of computer, you do not have the facilities of the Movie Toolbox available to you. If you want that application to exchange data with QuickTime applications on the Macintosh computer, you need to know the format of QuickTime movie resources.

Introduction to Movie Resources

Movie resources define the data interchange format for movies. The Movie Toolbox allows your application to create, view, edit, and store QuickTime movies. The functions of the Movie Toolbox shield your program from the details of how a movie is stored in the Macintosh file system (in the file type ‘Moov’). As a result, applications that run on Macintosh computers do not need to know anything about the internal structures of movie resources or movie files.

Storing Movies in Files

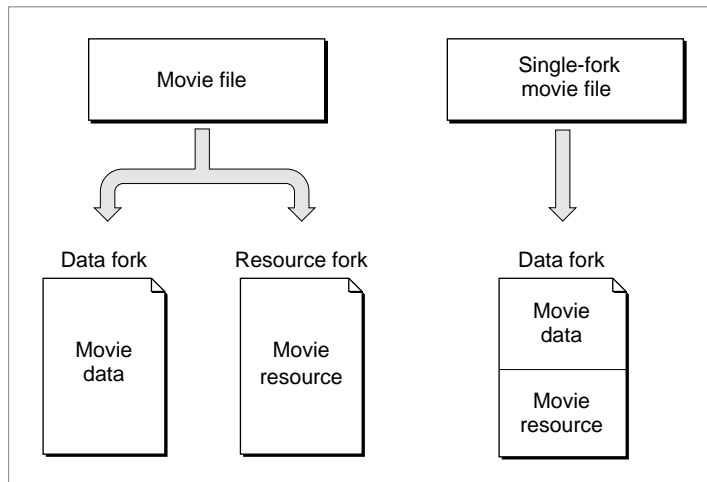
In the Macintosh file system, the Movie Toolbox typically uses both the resource fork and the data fork to store a QuickTime movie. The resource fork contains the movie resource. The data fork contains the actual movie data (or references to external data).

To facilitate data exchange between Macintosh computers and other computers, the Movie Toolbox can also understand movie files that store all the information for a movie in the data fork. These movie files are called **single-fork movie files**. Single-fork movie files are a possible way to export QuickTime movies to other systems, such as a computer using QuickTime for Windows.

Your application can create single-fork movie files from normal movie files by calling the Movie Toolbox's `FlattenMovieData` function (see the chapter "Movie Toolbox" for more information about this function). Your application can read single-fork movie files using the standard Movie Toolbox functions—you do not need to perform any special processing.

Figure 4-1 shows the difference between single-fork and normal movie files. A standard Macintosh movie file contains information in both the data and the resource forks. A single-fork movie file contains a data fork.

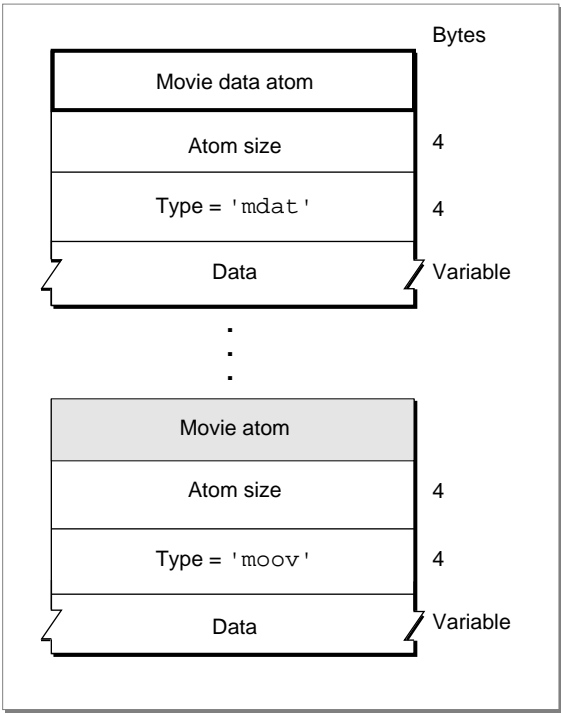
Figure 4-1 Movie files and single-fork movie files



Movie Resource Formats

Single-fork movie files store all the information for a movie in the data fork. The data fork contains two atoms: a movie data atom ('mdat ') and a movie resource atom. The movie's media data is stored in the movie data atom. Other atoms may follow the movie data atom. The movie resource atom follows the movie data atom and holds the description of the movie. There is no resource fork for this kind of movie file. Figure 4-2 shows the layout of a single-fork movie file. The movie data atom contains no other atoms, whereas the movie atom may contain other atoms.

Figure 4-2 The structure of a single-fork movie file



Atoms

The basic data unit in a QuickTime movie resource is the **atom**. Each atom contains size and type information along with its data. The `size` field indicates the number of bytes in the atom, including the `size` and `type` fields. The `type` field specifies the type of data stored in the atom and, by implication, the format of that data.

Atom Types

Table 4-1 lists the atom types defined by Apple and their corresponding constants and atom names.

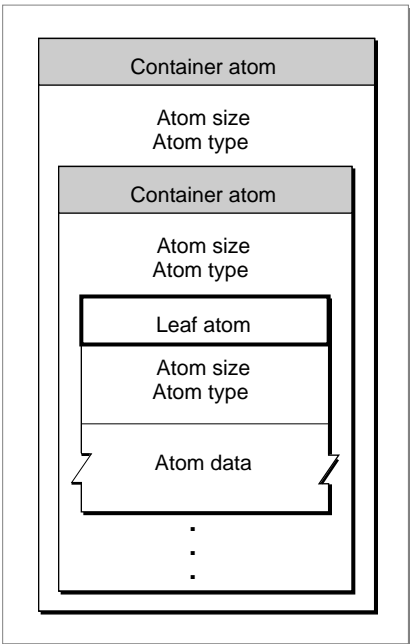
Table 4-1 Apple-defined atom types

| Constant | Atom type | Atom name |
|-------------------------|-----------|-------------------------------------|
| MovieAID | 'moov' | Movie atom |
| MovieHeaderAID | 'mvhd' | Movie header atom |
| ClipAID | 'clip' | Clipping atom |
| RgnClipAID | 'crgn' | Clipping region atom |
| MatteAID | 'matt' | Track matte atom |
| MatteCompAID | 'kmat' | Compressed matte atom |
| TrackAID | 'trak' | Track atom |
| UserDataAID | 'udta' | User-defined data atom |
| TrackHeaderAID | 'tkhd' | Track header atom |
| EditsAID | 'edts' | Edit atom |
| EditsListAID | 'elst' | Edit list atom |
| MediaAID | 'mdia' | Media atom |
| MediaHeaderAID | 'mdhd' | Media header atom |
| MediaInfoAID | 'minf' | Media information atom |
| VideoMediaInfoHeaderAID | 'vmhd' | Video media information header atom |
| SoundMediaInfoHeaderAID | 'smhd' | Sound media information header atom |
| DataInfoAID | 'dinf' | Data information atom |
| DataRefAID | 'dref' | Data reference atom |
| SampleTableAID | 'stbl' | Sample table atom |
| STSampleDescAID | 'stsd' | Sample description atom |
| STTimeToSampAID | 'stts' | Time-to-sample atom |
| STSyncSampleAID | 'stss' | Sync sample atom |
| STShadowSyncAID | 'stsh' | Shadow sync atom |
| STSampleToChunkAID | 'stsc' | Sample-to-chunk atom |
| HandlerAID | 'hdlr' | Handler reference atom |
| STSampleSizeAID | 'stsz' | Sample size atom |
| STChunkOffsetAID | 'stco' | Chunk offset atom |

The Layout of a QuickTime Atom

Figure 4-3 shows the layout of a sample QuickTime atom. Each atom carries its own size and type information as well as its data. Throughout this chapter, the name of a **container atom** (an atom that contains other atoms, including other container atoms) is printed across a horizontal gray band, and the name of a **leaf atom** (an atom that contains no other atoms) is printed across a horizontal drop shadow box. Leaf atoms contain data, usually in the form of tables.

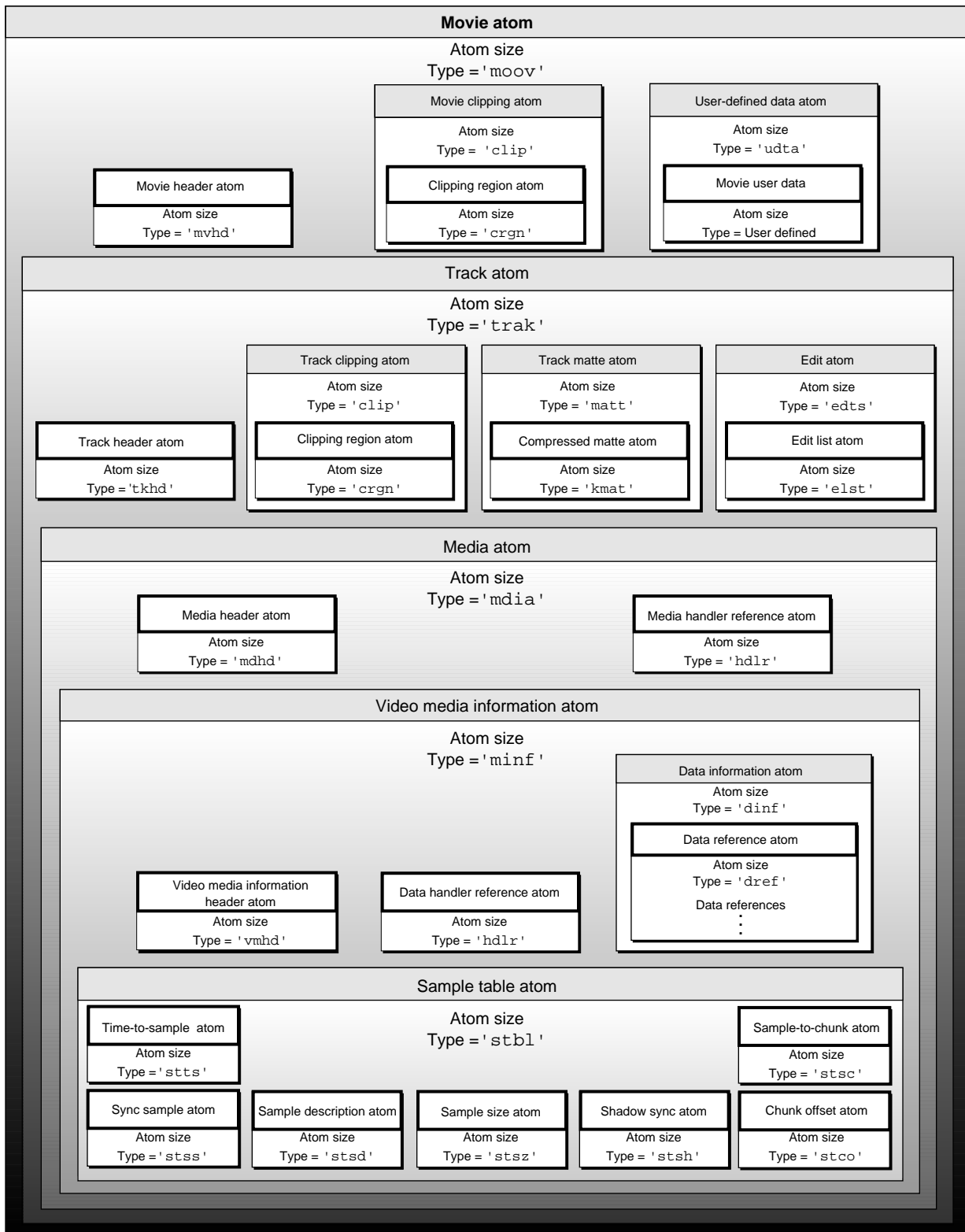
Figure 4-3 A sample QuickTime atom



Overview of the Movie Resource Atom

Movie resources consist of movie atoms, which in turn contain track atoms, which in turn contain media atoms (see the chapter “Movie Toolbox” in this book for information about the relationships between movies, tracks, and media structures). Leaf atoms usually contain tables of data. For example, the track atom contains the edit atom, which contains a leaf atom called the *edit list atom*. The edit list atom contains an edit list table. (See Figure 4-15 on page 4-25 for an illustration of the edit atom, and see Figure 4-16 on page 4-26 for an illustration of the edit list table.)

Figure 4-4 provides a conceptual view of the organization of a QuickTime movie, which, in this case, has one track containing video information. Each nested box in the illustration represents an atom that belongs to the atom underlying it. The figure does not show the data regions of any of the atoms. These areas are described in the pertinent sections that follow.

Figure 4-4 Sample organization of a one-track video movie

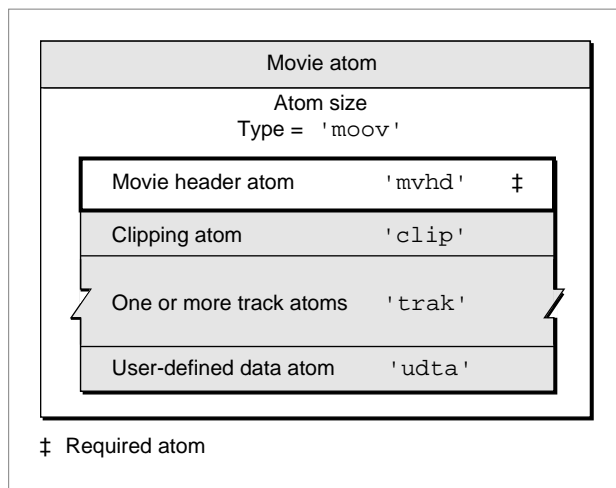
Movie Atoms

You can use movie atoms to specify information that defines a movie. 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.

The movie atom has an atom type of 'moov'. It contains other types of atoms, including one leaf atom—the movie header ('mvhd')—and several atoms that contain other atoms: a clipping atom ('clip'), one or more track atoms ('trak'), and user-defined data ('udta').

Figure 4-5 shows the layout of a movie atom. The movie header atom is the only required atom in the movie atom.

Figure 4-5 The layout of a movie atom



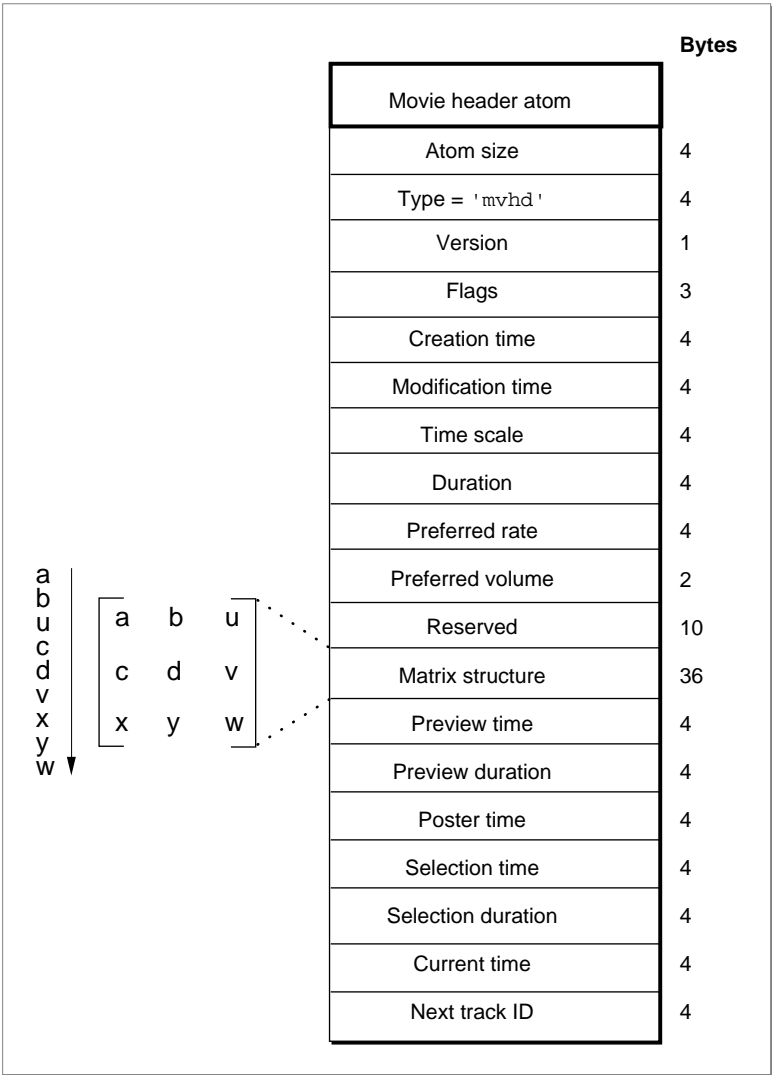
You define a movie atom by specifying these elements:

- **Size.** The number of bytes in this movie atom.
- **Type.** The type of this movie atom (the atom type, 'moov').
- **Movie header.** The movie header atom associated with this movie. See the next section for details on the movie header atom.
- **Movie clipping atom.** The clipping atom associated with this movie. See “Clipping Atoms,” which begins on page 4-23, for more information.
- **Track list.** One or more track atoms associated with this movie. See “Track Atoms,” which begins on page 4-14, for details on track atoms and their associated atoms.
- **User data.** The user-defined data atom associated with this movie. See “User-Defined Data Atoms,” which begins on page 4-20, for a discussion of user-defined data.

Movie Header Atoms

You can use the movie header atom to specify the characteristics of an entire movie. Figure 4-6 shows the layout of the movie header atom. The movie header atom is a leaf atom, which contains time information for the entire movie, such as time scale and duration. It also illustrates the data stream specified in the matrix structure field.

Figure 4-6 The layout of a movie header atom



Movie Resource Formats

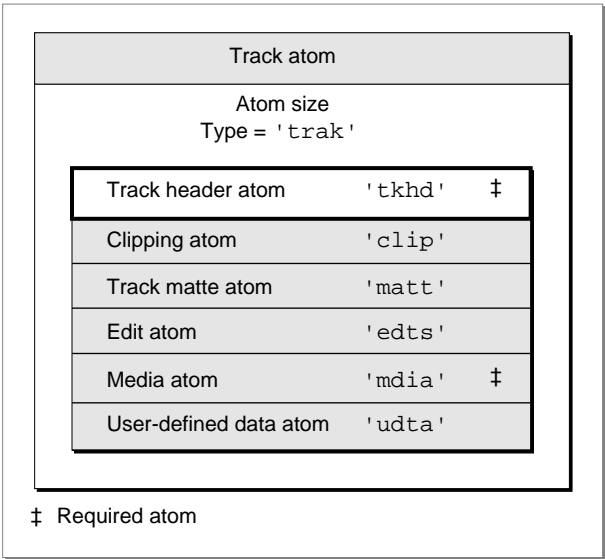
You define a movie header atom by specifying these elements:

- **Size.** A long integer that specifies the number of bytes in this movie header atom.
- **Type.** A long integer that specifies the format of the data in this movie header atom (defined by the atom type, 'mvhd').
- **Version.** A 1-byte specification of the version of this movie header atom.
- **Flags.** Three bytes of space for future movie header flags.
- **Creation time.** A long integer that specifies (in seconds since midnight, January 1, 1904) when the movie atom was created.
- **Modification time.** A long integer that specifies (in seconds since midnight, January 1, 1904) when the movie atom was changed.
- **Time scale.** A time value that indicates the **time scale** for this movie—that is, the number of time units that pass per second in its time coordinate system. A time coordinate system that measures time in sixtieths of a second, for example, has a time scale of 60.
- **Duration.** A time value that indicates the duration of the movie in time scale units.
- **Preferred rate.** A fixed number that specifies the rate at which to play this movie.
- **Preferred volume.** A 16-bit fixed number that specifies how loud to play this movie's sound.
- **Reserved.** Ten bytes reserved for use by Apple. Set to 0.
- **Matrix.** The matrix structure associated with this movie. A matrix shows how to map points from one coordinate space into another coordinate space. See the chapter "Movie Toolbox" in this book for details on matrix structures.
- **Preview time.** The time value in the movie at which the preview begins.
- **Preview duration.** The duration of the movie preview in movie time scale units. For more on time, see the chapter "Movie Toolbox" in this book.
- **Poster time.** The time value of the time of the movie poster.
- **Selection time.** The time value for the start time of the current selection.
- **Selection duration.** The duration of the current selection in movie time scale units.
- **Current time.** The time value for current time position within the movie.
- **Next track ID.** A long integer that indicates a value to use for the track ID number of the next track added to this movie.

Track Atoms

Track atoms define a single track of a movie. 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 its associated media atom. Figure 4-7 shows the layout of a track atom. The track atom requires the track header atom and the media atom.

Figure 4-7 The layout of a track atom



The track atom contains a track header atom ('tkhd'), a track clipping atom ('clip'), a track matte atom ('matt'), an edit atom ('edts'), a media atom ('mdia'), and a user-defined data atom ('udta'). You define a track atom by specifying these elements:

- Size. The number of bytes in this track atom.
- Type. The type of this track atom (the atom type, 'trak').
- Track header. The track header atom associated with this track. See the next section for details.
- Track clipping. The track clipping atom associated with this track. See "Clipping Atoms," which begins on page 4-23, for more on clipping atoms.
- Track matte. The track matte atom associated with this track. See "Track Matte Atoms," which begins on page 4-24, for more on track matte atoms.
- Edits. The edit atom associated with this track. See "Edit Atoms," which begins on page 4-25, for details.

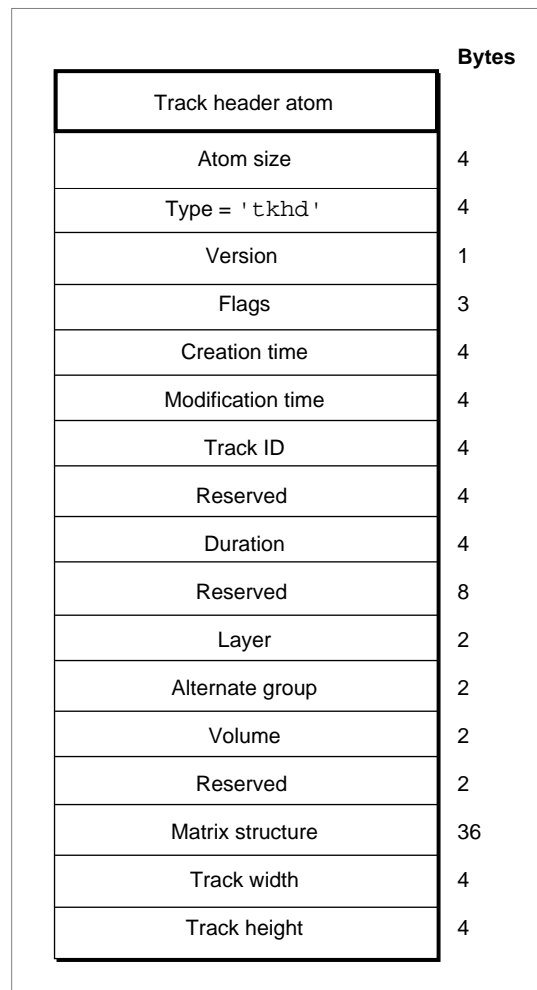
Movie Resource Formats

- **Media.** The media atom associated with this track. See “Media Atoms,” which begins on page 4-17, for details.
- **User data.** The user-defined data atom associated with this track. This field is used for extension with new data types. See “User-Defined Data Atoms,” which begins on page 4-20, for details.

Track Header Atoms

The track header atom specifies the characteristics of a single track within a movie. A track header atom contains a `size` field that specifies the number of bytes and a `type` field that indicates the format of the data (defined by the atom type, 'tkhd'). Figure 4-8 shows the structure of the track header atom.

Figure 4-8 The layout of a track header atom



Movie Resource Formats

The track header atom contains the track characteristics for the track, including temporal, spatial, and volume information. You define a track header atom by specifying these elements:

- **Size.** A long integer that specifies the number of bytes in this track header atom.
- **Type.** A long integer that specifies the type of data in this track header atom (defined by the atom type, 'tkhd').
- **Version.** A 1-byte specification of the version of this track header.
- **Track header flags.** Three bytes that are reserved for the track header flags, which adjust the remaining fields in the track header according to the kind of movie track you specify with the following enumeration:

```
enum
{
    TrackEnable = 1<<0, /* enabled track */
    TrackInMovie = 1<<1, /* track in playback */
    TrackInPreview = 1<<2, /* track in preview */
    TrackInPoster = 1<<3 /* track in poster */
};
```

- **Creation time.** A long integer that indicates (in seconds since midnight, January 1, 1904) when the track header was created.
- **Modification time.** A long integer that indicates (in seconds since midnight, January 1, 1904) when the track header was changed.
- **Track ID.** A long integer that specifies the value to use for the track ID number.
- **Reserved.** A long integer that is reserved for use by Apple. Set the value of this field to 0.
- **Duration.** The duration of this track (in movie time).
- **Reserved.** An 8-byte value that is reserved for use by Apple. Set the value of this field to 0.
- **Layer.** The priority of playing this track in a movie. When it plays a movie, the Movie Toolbox displays the movie's tracks according to their **layer**—tracks with lower layer numbers are displayed in front; tracks with higher layer numbers are displayed in back.
- **Alternate group.** A short integer that specifies a collection of movie tracks that contain alternate data for one another. QuickTime chooses one track from the group to be used when the movie is played. The choice may be based on such considerations as playback quality or language and the capabilities of the computer.
- **Volume.** A short integer that indicates how loudly this track's sound is to be played.
- **Reserved.** A short integer that is reserved for use by Apple. Set the value of this field to 0.
- **Matrix.** The matrix structure associated with this track. See Figure 4-6 on page 4-12 for an illustration of a matrix structure.

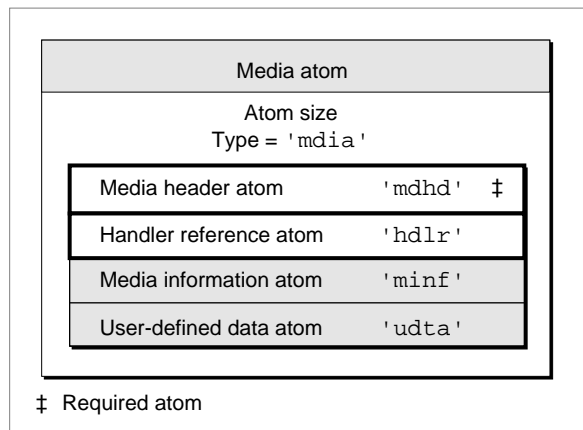
Movie Resource Formats

- Track width. A fixed number that specifies the width of this track.
- Track height. A fixed number that indicates the height of this track.

Media Atoms

Media atoms define the data for a movie track. The media atom contains information that specifies the component that is to interpret the media data, and it also specifies the data references. Figure 4-9 shows the layout of a media atom.

Figure 4-9 The layout of a media atom



The media atom has an atom type of 'mdia'. It may contain other atoms, such as a media header ('mdhd'), a handler reference ('hdlr'), media information ('minf'), and user-defined data ('udta'). The only required atom in a media atom is the media header atom.

Note

The handler reference atom lets you know what kind of media this media atom contains—for example, video or sound. The layout of the media information atom is specific to the media handler that is to interpret the media. “Media Information Atoms,” which begins on page 4-27, discusses how data may be stored in a media, using the video media format defined by Apple as an example. ♦

You define a media atom by specifying these elements:

- Size. A long integer that specifies the number of bytes in this media atom.
- Type. A long integer that specifies the type of the data in this media atom (defined by the 'mdia' atom type).
- Media header. The media header atom, which is described in the next section. It contains the standard media information.

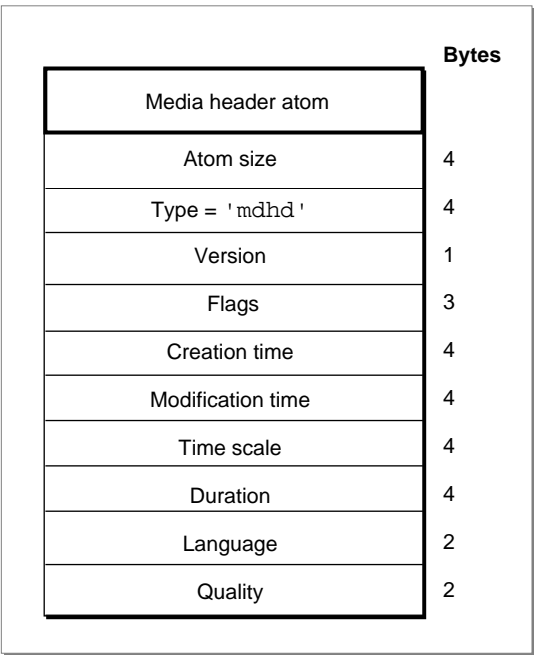
Movie Resource Formats

- Media handler. The media handler, which is defined by the handler reference atom, described in “Handler Reference Atoms,” which begins on page 4-19.
- Media information. The media information atom. For an example of a media information atom, see “Video Media Information Atoms,” which begins on page 4-27.
- User data. The user-defined data atom associated with this media. This field is used for extension with new data types. See “User-Defined Data Atoms,” which begins on page 4-20, for details.

Media Header Atoms

The media header atom specifies the characteristics of the media that is used to store data for the movie track defined in its associated track atom. The media header atom contains the number of bytes in the media header atom, the format of the data in the media header atom (defined by the 'mdhd' atom type), and the media header. The media characteristics include temporal information. Figure 4-10 shows the layout of the media header atom.

Figure 4-10 The layout of a media header atom



You define a media header atom by specifying these elements:

- Size. A long integer that specifies the number of bytes in this media header atom.
- Type. A long integer that specifies the type of data in this media header atom (defined by the atom type, 'mdhd').

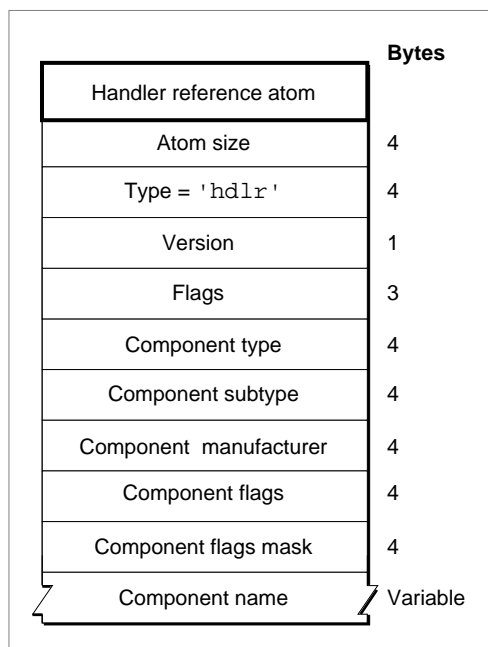
Movie Resource Formats

- **Version.** One byte that specifies the version of this movie.
- **Flags.** Three bytes of space for future media header flags.
- **Creation time.** A long integer that specifies (in seconds since midnight, January 1, 1904) when the media atom was created.
- **Modification time.** A long integer that specifies (in seconds since midnight, January 1, 1904) when the media atom was changed.
- **Time scale.** A time value that indicates the time scale for this media—that is, the number of time units that pass per second in its time coordinate system.
- **Duration.** The duration of this media in media time scale units.
- **Language.** A short integer that specifies the language code for this media. See *Inside Macintosh: Text* for more on language codes.
- **Quality.** A short integer that specifies the playback quality (that is, its suitability for playback in a given environment) for this media. See the chapter “Movie Toolbox” in this book for details on playback quality.

Handler Reference Atoms

The handler reference atom specifies the component that is to interpret a media’s data. This component is called a *media handler*. (See the chapter “Component Manager” in *Inside Macintosh: More Macintosh Toolbox* for more information about components.) Figure 4-11 shows the layout of a handler reference atom.

Figure 4-11 The layout of a handler reference atom



Movie Resource Formats

You define a handler reference atom by specifying these elements:

- **Size.** The number of bytes in this handler reference atom.
- **Type.** The type of the data (defined by the 'hdlr' atom type) in the handler reference atom.
- **Flags.** A 1-byte specification of the version of this handler information.
- **Version.** A 3-byte space for future handler information flags.
- **Component type.** A four-character code that identifies the type of the media handler. All components of a particular type must support a common set of functions. Examples of component types are 'mhlr' and 'dhlr'.
- **Component subtype.** A four-character code that identifies the type of the media handler. Different types of a component type may support additional features or provide interfaces that extend beyond the standard functions for a given component type value. For media handlers, this field defines the type of data—for example, 'vide' or 'soun'.
- **Component manufacturer.** A four-character code that identifies the manufacturer of this media handler. This field allows for further differentiation between individual components. For example, components made by a specific manufacturer may support an extended feature set.
- **Component flags.** A 32-bit field that provides additional information about a particular media handler. The most significant 8 bits are reserved for use by the Component Manager and provide both static and dynamic information about the component.
- **Component flags mask.** A 32-bit field that indicates which flags in the component mask are relevant to a particular search operation. These flags are used when searching for a handler component.
- **Component name.** A Pascal string that specifies the name of the component—that is, the original handler used when this movie was created.

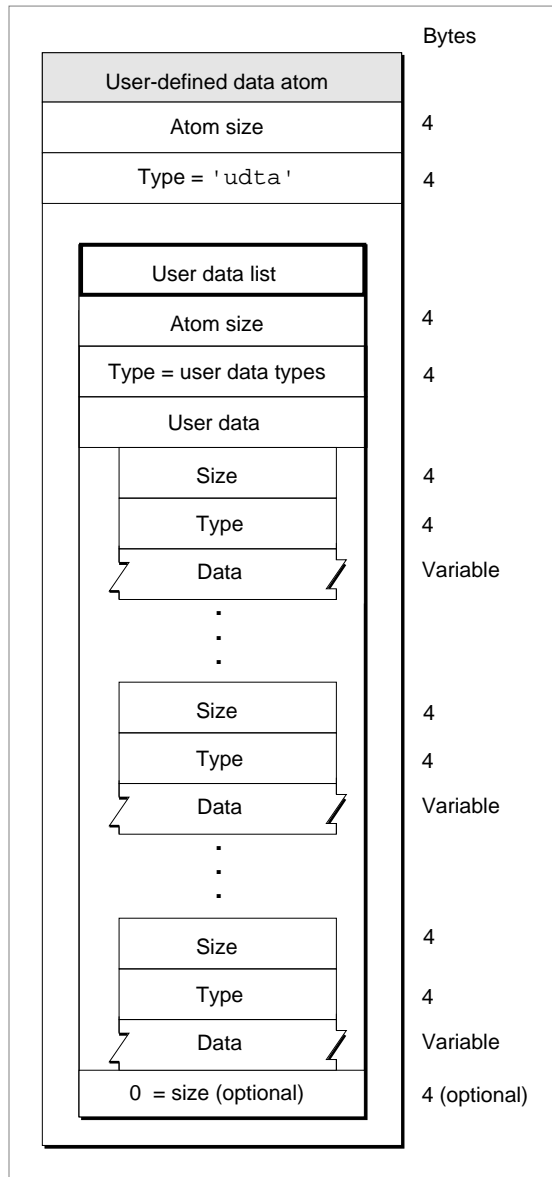
User-Defined Data Atoms

Many movie, track, and media atoms contain atoms that store user-defined data. Your application may store data in these user-defined data atoms.

Movie Resource Formats

Figure 4-12 shows the layout of a user-defined data atom.

Figure 4-12 The layout of a user-defined data atom



Movie Resource Formats

You define a user-defined data atom by specifying these elements:

- Size. The number of bytes in the data element.
- Type. The type of the data in the data element (defined by the 'udta' atom type).
- User data list. The movie user data atom contains a user data list that is itself formatted like a series of atoms. Each data element in the private data portion of the user-defined data atom contains size and type information along with the data. Furthermore, the list of atoms is optionally terminated by a 0.

The following user data types are currently defined:

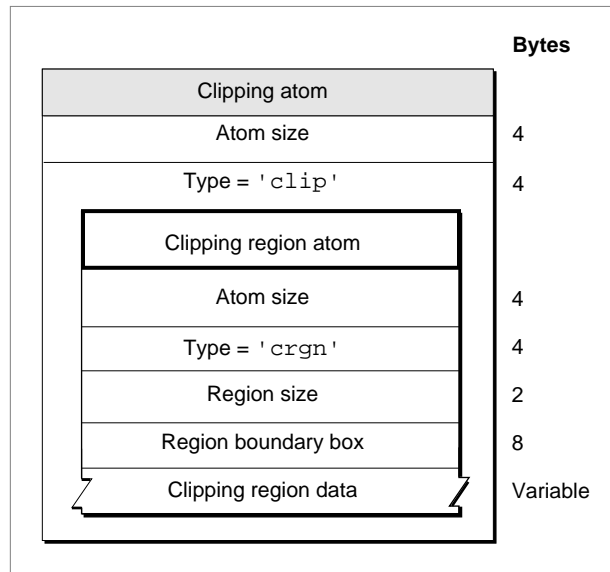
| | |
|------------------|---|
| '@cpy' | Copyright statement |
| '@day' | Date the movie content was created |
| '@dir' | Name of movie's director |
| '@ed1' to '@ed9' | Edit dates and descriptions |
| '@fmt' | Indication of movie format (computer-generated, digitized, and so on) |
| '@inf' | Information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

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

Clipping Atoms

Clipping atoms specify the clipping regions for movies and for tracks. Figure 4-13 shows the layout of clipping atoms.

Figure 4-13 The layout of a clipping atom



You define a clipping atom by specifying these elements:

- Size. The number of bytes in this clipping atom.
- Type. The type of the data in this clipping atom (defined by the 'clip' atom type).
- Clipping region atom. Described in the next section.

Clipping Region Atoms

The clipping region atom specifies the clipping data. The layout of the clipping region atom is shown in Figure 4-13. You define a clipping region atom by specifying these elements:

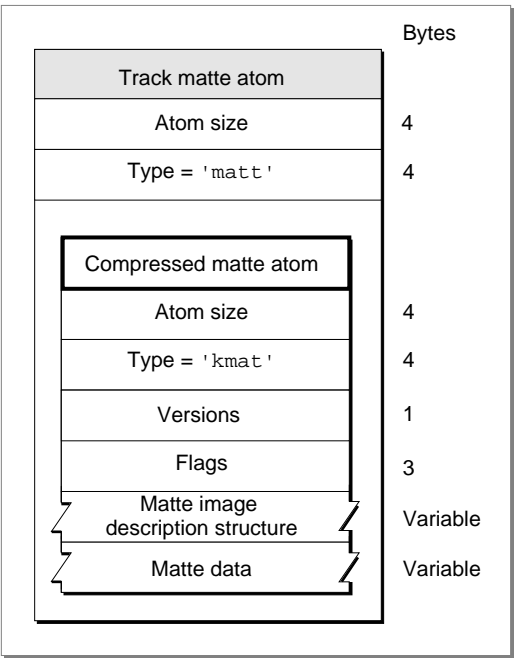
- Size. A long integer that indicates the number of bytes in the clipping region data atom.
- Type. A long integer that indicates the type of the clipping region data (defined by the 'crgn' atom type).

The region size, region boundary box, and data fields constitute a QuickDraw region. See the chapter “QuickDraw” in *Inside Macintosh: Imaging* for details on QuickDraw regions.

Track Matte Atoms

Track matte atoms specify the mattes for tracks. (A **track matte** is a pixel map that defines the blending of visual track data. See the chapter “Movie Toolbox” in this book for details.) Figure 4-14 shows the layout of track matte atoms.

Figure 4-14 The layout of a track matte atom



You define a track matte atom by specifying these elements:

- Size. A long integer that specifies that number of bytes in this track matte atom.
- Type. A long integer that specifies the type of this track matte atom (defined by the 'mat t' atom type).
- Compressed matte atom. The compressed matte atom, which is described in the next section.

Compressed Matte Atoms

The compressed matte atom specifies the image description structure associated with a particular matte atom. The layout of the compressed matte atom is shown in Figure 4-14.

Movie Resource Formats

You define a compressed matte atom by specifying these elements:

- Size. A long integer that indicates the number of bytes in this compressed matte atom.
- Type. A long integer that indicates the type of the data in this atom (defined by the 'kmat' atom type).
- Version. A 1-byte specification of the version of this compressed matte atom.
- Flags. Three bytes of space for future flags associated with this compressed matte atom.
- Matte image description. An image description structure of variable length and associated with this matte data. See the chapter “Image Compression Manager” in this book for details on the image description structure.
- Matte data. The compressed matte data, which is of variable length.

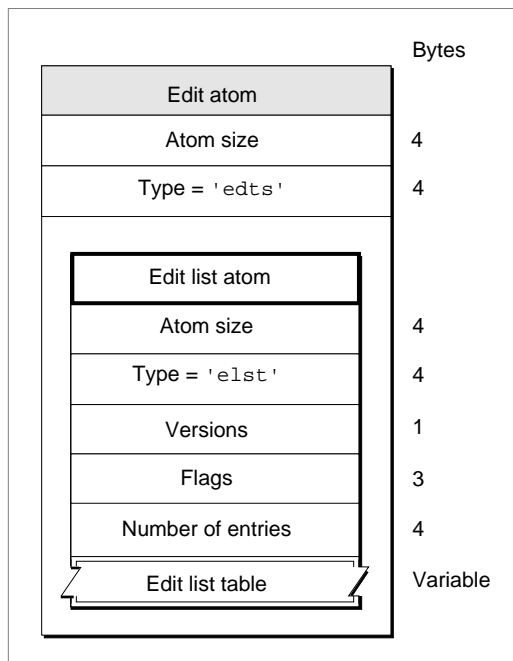
Edit Atoms

You can use edit atoms to define the portions of the media that are to be used to build up a track for a movie. Figure 4-15 shows the layout of an edit atom.

Note

If the edit atom or the edit list atom is missing, you can assume that the entire media is contained in the track. ♦

Figure 4-15 The layout of an edit atom



You define an edit atom by specifying these elements:

- Size. A long integer that indicates the number of bytes in this edit atom.
- Type. A long integer that indicates the type of the data in this edit atom (defined by the 'edts' atom type).
- Edit list. The edit list atom that contains the edit list information, described in the next section.

Edit List Atoms

You can use the edit list atom, also shown in Figure 4-15, to tell QuickTime how to map from a time in a movie to a time in a media, and ultimately to media data. This information is in the form of an edit list table, shown in Figure 4-16.

You define an edit list atom by specifying the following elements:

- Size. A long integer that specifies the number of bytes in the edit list atom.
- Type. A long integer that specifies the type of the edit list data (defined by the 'elst' atom type).
- Version. A 1-byte specification of the version of this edit list atom.
- Flags. Three bytes of space for future flags to be associated with this edit list atom.
- Number of entries. The number of entries in the edit list atom.
- Edit list table. Each entry in the edit list table (shown in Figure 4-16) describes a single edit and contains a track duration field, a media time field, and a media rate field.

Figure 4-16 The layout of an edit list table

| | | | |
|-----------------|------------|------------|-------|
| Edit list table | | | |
| Track duration | Media time | Media rate | Field |
| 4 | 4 | 4 | Bytes |

You create an edit list table by specifying these elements:

- Track duration. The duration of this edit segment in movie time scale units.
- Media time. The starting time within the media of this edit segment (in media time scale units). If -1, it is an empty edit.
- Media rate. A fixed number that specifies the relative rate at which to play the media for this edit segment.

Media Information Atoms

Media information atoms (defined by the 'minf' data type) 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. These atoms are formatted differently based on the type of media data stored in the atom. 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. This section describes examples of atoms that store media information for the video (defined by the 'vmhd' atom type) and sound (defined by the 'smhd' atom type) portions of QuickTime movies.

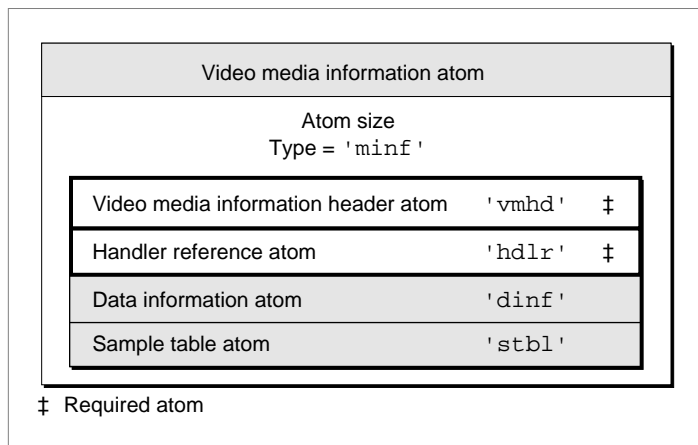
Note

“Using Media Information Atoms,” which begins on page 4-46, discusses how the video media handler locates samples in a video media. ♦

Video Media Information Atoms

Video media information atoms are the highest-level atoms in video media. A number of other atoms define specific characteristics of the video media data. Figure 4-17 shows the layout of a video media information atom.

Figure 4-17 The layout of a media information atom for video



You define a video media information atom by specifying these elements:

- **Size.** A long integer that specifies the number of bytes in this video media information atom.

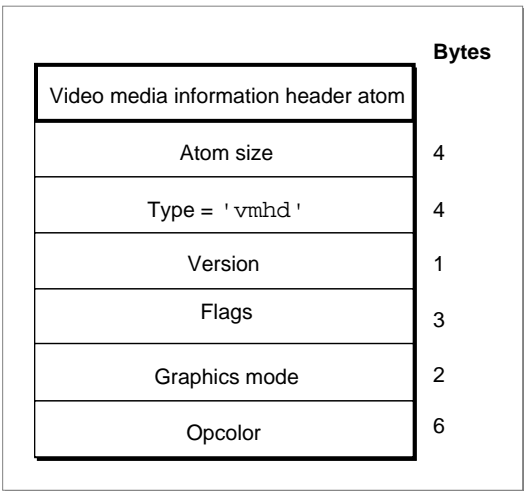
Movie Resource Formats

- **Type.** A long integer that specifies the type of the data (defined by the 'minf' atom type) in this media information header.
- **Video media information.** The video media information header atom (a required atom), which is described in the next section.
- **Handler reference.** The handler reference atom (a required atom), which contains information specifying the data handler component that provides access to the media data. See the chapter “Component Manager” in *Inside Macintosh: More Macintosh Toolbox* for more information about components. Figure 4-11 on page 4-19 shows the layout of a handler reference atom. The handler reference uses the data information atom, described by the `datainfo` field in the video media information structure.
- **Data information.** The data information atom, described in “Data Information Atoms” on page 4-31.
- **Sample table.** The sample table atom, described in “Sample Table Atoms” on page 4-34.

Video Media Information Header Atoms

Video media information atoms are the highest-level atoms in video media. A number of other atoms define specific characteristics of the video media data. Figure 4-18 shows the structure of a video media information header atom.

Figure 4-18 The layout of a media information header atom for video



You define a video media information header atom by specifying these elements:

- **Size.** A long integer that specifies the number of bytes in the media information in this video media information header.

Movie Resource Formats

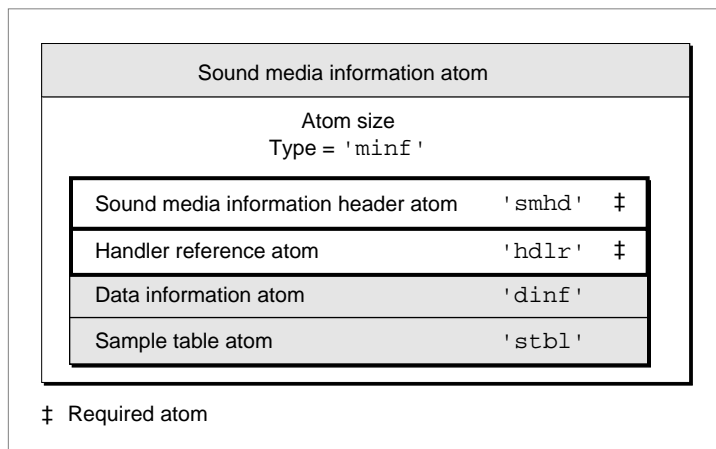
- **Type.** A long integer that specifies the type of the data (defined by the 'vmhd' atom type) in this video media information header.
- **Version.** A 1-byte specification of the version of this video media information header.
- **Flags.** A 3-byte space for video media information flags. The `videoFlagNoLeanAhead` flag is available, which instructs QuickTime that the video was not created skewed and that it should use a technique having greater accuracy.
- **Graphics mode.** A short integer that specifies the transfer mode, which is a specification of which Boolean operation QuickDraw should perform when drawing or transferring an image from one location to another.
- **Opcolor.** Three 16-bit values that specify the red, green, and blue colors for the transfer mode operation indicated in the graphics mode field.

For comprehensive details on QuickDraw's transfer modes and opcolors and their values, see *Inside Macintosh: Imaging*.

Sound Media Information Atoms

Sound media information atoms are the highest-level atoms in sound media. These atoms define specific characteristics of the sound media data. Figure 4-19 shows the layout of a sound media information atom.

Figure 4-19 The layout of a media information atom for sound



In addition to the size and type information, the sound media information atom contains the sound media information header atom, which is described in the next section, and the handler reference atom, the data information atom, and the sample table atom.

Movie Resource Formats

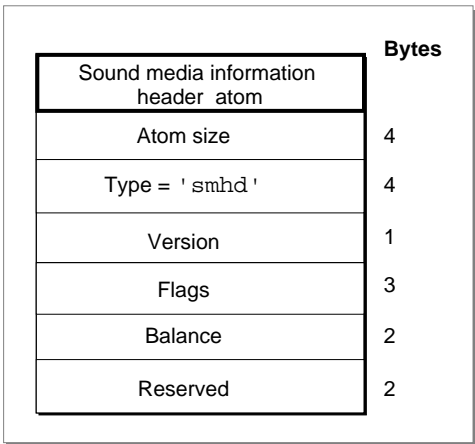
You define a sound media information atom by specifying these elements:

- Size. A long integer that specifies the number of bytes in this sound media information atom.
- Type. A long integer that specifies the type of the data in this sound media information header (defined by the 'minf' data type).
- Sound media information. The sound media information header atom (a required atom), which is described in the next section.
- Handler reference. The handler reference atom (a required atom), which contains information specifying the data handler component that provides access to the media data. See the chapter “Component Manager” in *Inside Macintosh: More Macintosh Toolbox* for more information about components. Figure 4-11 on page 4-19 shows the layout of a handler reference atom. The handler reference atom uses the data information atom, described by the `dataInfo` field in this sound media information structure.
- Data information. The data information atom, described in “Data Information Atoms,” which begins on page 4-31.
- Sample table. The sample table atom, described in “Sample Table Atoms,” which begins on page 4-34.

Sound Media Information Header Atoms

The sound media information header atom (shown in Figure 4-20) stores the sound media information.

Figure 4-20 The layout of a sound media information header atom



Movie Resource Formats

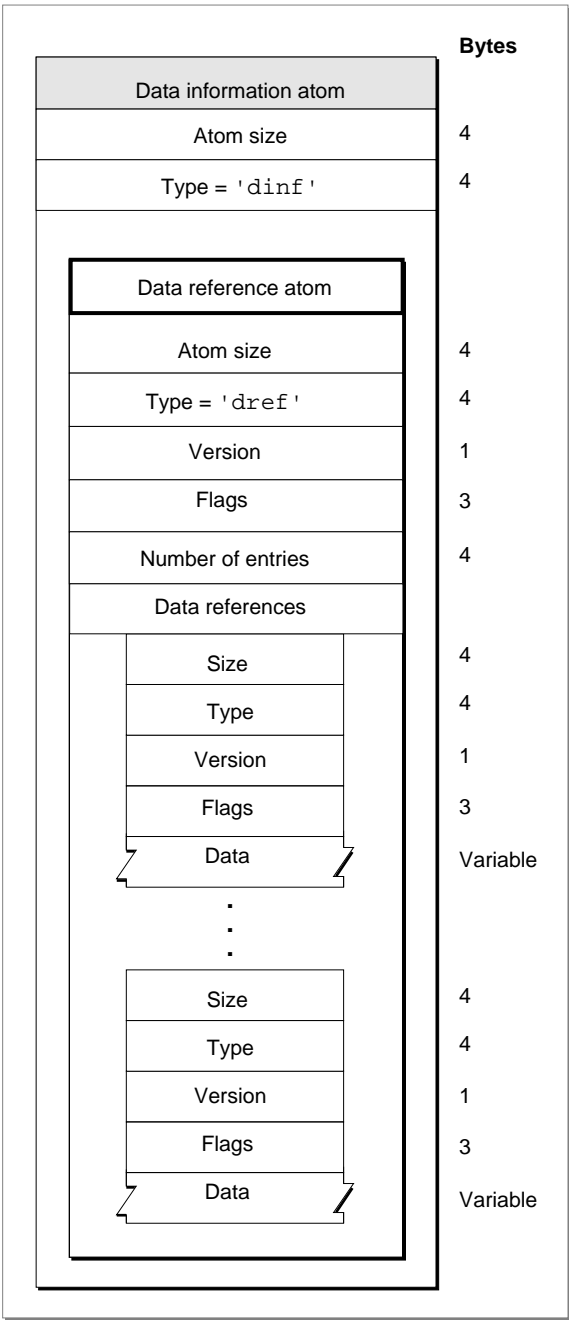
You define a sound media information header atom by specifying these elements:

- **Size.** A long integer that specifies the number of bytes in this sound media information header atom.
- **Type.** A long integer that specifies the type of the data in this sound media information header atom (defined by the 'smhd' data type).
- **Version.** A 1-byte specification of the version of this sound media information header.
- **Flags.** Three bytes of space for future associated flags.
- **Balance.** A short integer that specifies the sound balance of this sound media. (Sound balance is the setting that controls the mix of sound between the two speakers of a computer.) This field is normally set to 0. See the chapter “Movie Toolbox” in this book for more on sound balance.
- **Reserved.** Reserved for use by Apple. Set this field to 0.

Data Information Atoms

The handler reference atom (described in “Handler Reference Atoms,” which begins on page 4-19) contains information specifying the data handler component that provides access to the media data. See the chapter “Component Manager” in *Inside Macintosh: More Macintosh Toolbox* for more about components. The handler uses the data information atom, which you can use to specify where the media data is stored. Figure 4-21 shows the layout of the data information atom.

Figure 4-21 The layout of a data information atom



You define a data information atom by specifying these elements:

- **Size.** A long integer that specifies the number of bytes in this data information atom.
- **Type.** A long integer that specifies the format (defined by the 'dinf' atom type) of the data in this data information atom.

Movie Resource Formats

- Data references. The data reference atom, described in the next section, contains the data references.

Data Reference Atoms

Figure 4-21 also shows the data reference atom, which encompasses the data references.

You define a data reference atom by specifying these elements:

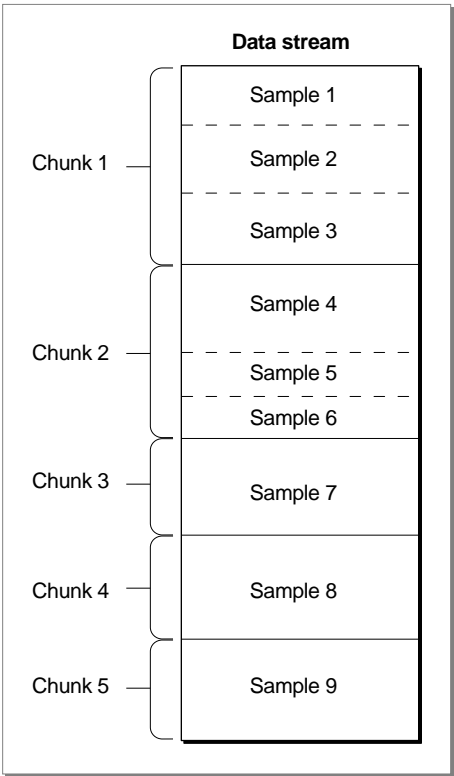
- Size. A long integer that specifies the number of bytes in this data reference container atom.
- Type. A long integer that specifies the type of the data in the data reference atom (defined by the 'dref' data type).
- Version. A 1-byte specification of the version of this data reference atom.
- Flags. Three bytes that contain space for future flags.
- Number of entries. A count of entries in the data references field.
- Data references. Data references are formatted like atoms, as follows:
 - Size. A long integer that specifies the number of bytes in these data references.
 - Type. A long integer that specifies the type of the data (currently defined by the 'alis' data type on the Macintosh computer) in the data references.
 - Version. A 1-byte specification of the version of these data references.
 - Flags. Three bytes that contain the attributes of the data in these data references. One enumerated constant is available. The `dataRefSelfReference` attribute denotes that the data comes from the same location as the movie resource. If the movie resource came from a resource fork, the movie data is in the data fork of the same file. In the case of a single-fork file, the movie data is also in the data fork of the file.
 - Data references. The data reference information. (For the current data handlers, this is an alias).

An Introduction to Samples

One way to describe a sample (that is, a single element of a sequence of time-ordered data) is to include it in a sample table atom. Samples are stored sequentially in the media, and they may have varying durations. This approach enforces an ordering of the samples—it does not mean the sample data must be stored sequentially with respect to movie time in the actual data stream. Figure 4-22 shows the way that samples are stored in a series of **chunks** in a media. Chunks are a collection of data samples in a media that

allow 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.

Figure 4-22 Samples in a media



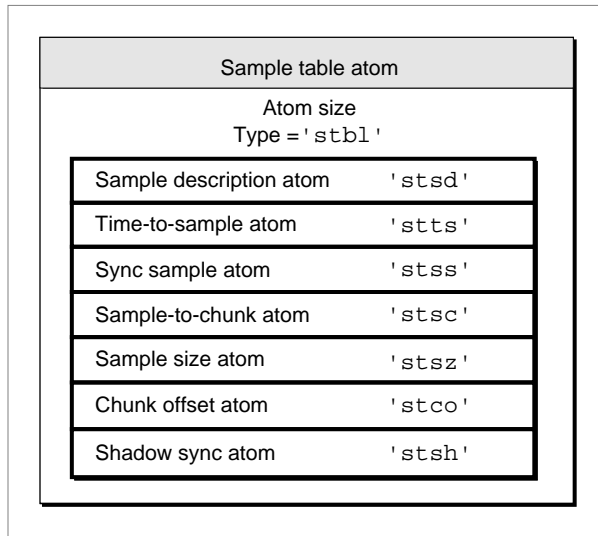
Sample Table Atoms

The sample table atom contains information for converting from media time to sample number to sample location. This atom also indicates how to interpret the sample (for example, whether to decompress the video sample and, if so, how). This section describes the format and content of the sample table atom.

The sample table has an atom type of 'stbl'. It contains the sample description atom, the time-to-sample atom, the sample-to-chunk atom, the sync sample atom, the sample size atom, the chunk offset atom, and the shadow sync atom.

Figure 4-23 shows the layout of the sample table atom.

Figure 4-23 The layout of a sample table atom



You define a sample table atom by specifying these elements:

- **Size.** A long integer that specifies the number of bytes in the sample table atom.
- **Type.** A long integer that specifies the type of the data (defined by the 'stbl' atom type) in the sample table atom.
- **Sample description.** The sample description atom, described in the next section.
- **Time-to-sample.** The time-to-sample atom, described in “Time-to-Sample Atoms,” which begins on page 4-37.
- **Sync sample.** The sync sample atom, described in “Sync Sample Atoms,” which begins on page 4-39.
- **Sample-to-chunk.** The sample-to-chunk atom, described in “Sample-to-Chunk Atoms,” which begins on page 4-40.
- **Sample size.** The sample size atom, described in “Sample Size Atoms,” which begins on page 4-42.
- **Chunk offset.** A chunk offset atom, described in “Chunk Offset Atoms,” which begins on page 4-43.
- **Shadow sync.** The shadow sync atom, described in “Shadow Sync Atoms,” which begins on page 4-45.

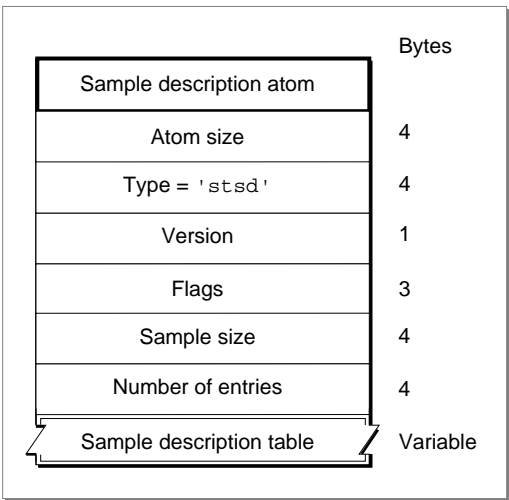
The following sections discuss each of the atoms that may be contained in a sample table.

Sample Description Atoms

The sample description atom stores information for the decoding of samples in the media. In the case of video media, the sample descriptions are image description structures (see the chapter “Image Compression Manager” earlier in this book for more information about image descriptions). Figure 4-24 shows the layout of the sample description atom.

The sample description atom has an atom type of 'stsd'. The sample description atom contains a table of sample descriptions, each of which contains a single sample description. A media may have one or more sample descriptions, depending upon the number of different compression types used in the media. The sample-to-chunk atom identifies the sample description for each sample in the media by specifying the index into this table for the appropriate description (see “Sample-to-Chunk Atoms,” which begins on page 4-40).

Figure 4-24 The layout of a sample description atom



You define a sample description atom by specifying these elements:

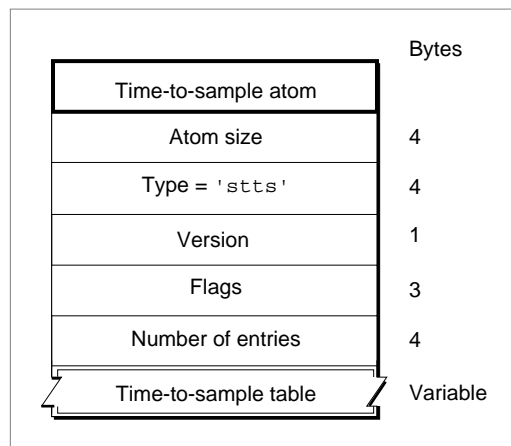
- Size. A long integer that specifies the number of bytes in this sample description atom.
- Type. A long integer that specifies the type (defined by the atom type 'stsd') of the data in this sample description atom.
- Version. A 1-byte specification of the version number of this sample description atom.
- Flags. Three bytes of space for future flags associated with it.
- Number of entries. A long integer that specifies how many entries in the sample description table are listed in the sample description table field of this atom.
- Sample description table. The sample description table, which contains a list of sample descriptions.

Time-to-Sample Atoms

Time-to-sample atoms store duration information for the samples in a media, providing a mapping from a time in a media to the corresponding data sample. The time-to-sample atom has an atom type of 'stts'.

You can determine the appropriate sample for any given time in a media by examining the time-to-sample atom (shown in Figure 4-25), which contains the time-to-sample atom table.

Figure 4-25 The layout of a time-to-sample atom



You define a time-to-sample atom by specifying these elements:

- **Size.** A long integer that specifies the number of bytes in this time-to-sample atom.
- **Type.** A long integer that specifies the type (defined by the 'stts' atom type) of the data contained in the time-to-sample atom.
- **Version.** A 1-byte specification of the version number of this time-to-sample atom.
- **Flags.** Three bytes of space for any future flags associated with this time-to-sample atom.
- **Number of entries.** A long integer that specifies the number of entries in the time-to-sample table.
- **Time-to-sample table.** The time-to-sample atom contains a table that defines the duration of each sample in the media. Each table entry contains a count field and a duration field. The structure of the time-to-sample table is shown in Figure 4-26.

Figure 4-26 The layout of a time-to-sample table

| | |
|----------------------|-----------------|
| Time-to-sample table | |
| Sample count | Sample duration |
| 4 | 4 |

Field
Bytes

You define a time-to-sample table by specifying these entries:

- Sample count. A long integer that specifies the number of consecutive samples that have the same duration.
- Sample duration. A long integer that specifies the duration of each sample.

Entries in the table 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 has a constant frame rate, this table would have one entry.

Figure 4-27 shows an example of a time-to-sample table that is based on the data stream shown in Figure 4-22 on page 4-34. Figure 4-22 shows a total of nine samples that correspond in count and duration to the entries of the table shown in Figure 4-27. Even though samples 4, 5, and 6 are in the same chunk, sample 4 has a duration of 3, and samples 5 and 6 have a duration of 2.

Figure 4-27 An example of a time-to-sample table

| | | |
|----------------------|---|---|
| Time-to-sample table | | |
| 4 | 2 | 3 |
| 3 | 1 | 2 |

Sample count
Sample duration

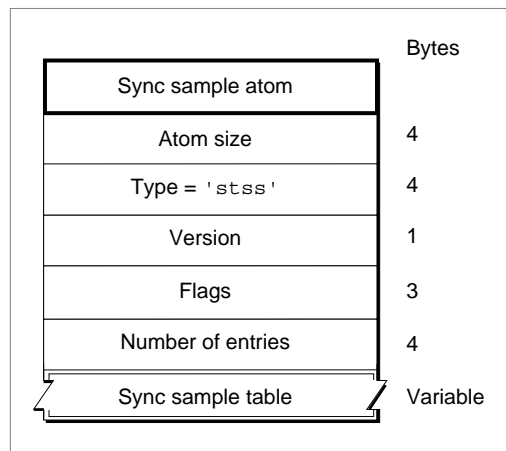
Sync Sample Atoms

The sync sample atom identifies the **key frames** in the media. In a media that contains compressed data, key frames define starting points for portions of a temporally compressed sequence (see the chapter “Image Compression Manager” in this book for more information about key frames and temporal compression in video data). The key frame is self-contained—that is, it is independent of preceding frames. Subsequent frames may depend on the key frame.

Sync sample atoms have an atom type of 'stss'. The sync sample atom contains a table of sample numbers. Each entry in the table identifies a sample that is a key frame for the media. Figure 4-28 shows the layout of a sync sample atom.

If no sync sample atom exists, then all the samples are key frames.

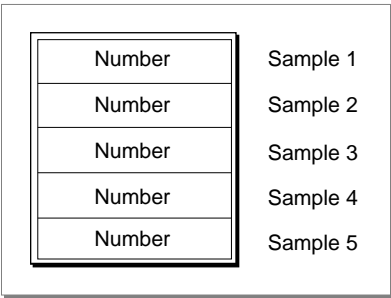
Figure 4-28 The layout of a sync sample atom



You define a sync sample atom by specifying these elements:

- **Size.** A long integer that specifies the number of bytes in this sync sample atom.
- **Type.** A long integer that specifies the type of the data of this sync sample atom (defined by the 'stss' atom type).
- **Version.** A 1-byte specification of the version of this sync sample atom.
- **Flags.** Three bytes of space for future flags.
- **Number of entries.** A long integer that specifies how many sample numbers are in the sync sample table contained in the sync sample table field.
- **Sync sample table.** The sync sample table (shown in Figure 4-29) consists of an array of sample numbers. Each entry in the table identifies a sample that is a key frame for the media.

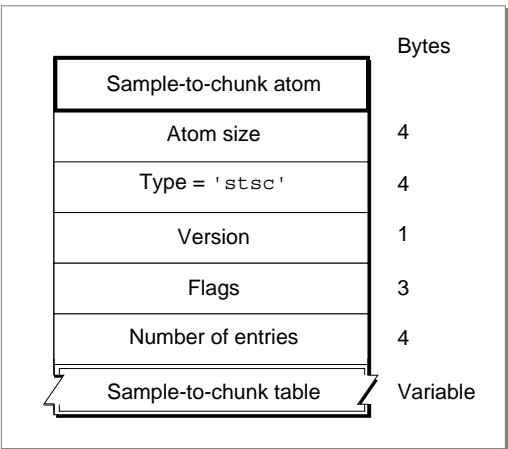
Figure 4-29 The layout of a sync sample table



Sample-to-Chunk Atoms

As samples are added to a media, they are collected into chunks that allow 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. The sample-to-chunk atom stores chunk information for the samples in a media. Figure 4-30 shows the layout of the sample-to-chunk atom. By examining the sample-to-chunk atom, you can determine the chunk that contains a specific sample.

Figure 4-30 The layout of a sample-to-chunk atom



You define a sample-to-chunk atom by specifying these elements:

- Size. A long integer that specifies the number of bytes in this sample-to-chunk atom.
- Type. A long integer that specifies the type of the data in this sample-to-chunk atom (defined by the 'stsc' atom type).
- Version. A 1-byte specification of the version of this sample-to-chunk atom.
- Flags. Three bytes of space for future flags associated with this sample-to-chunk atom.

Movie Resource Formats

- Number of entries. The number of entries in the sample-to-chunk table.
- Sample-to-chunk table. Figure 4-31 shows the structure of a sample-to-chunk table. Each sample-to-chunk atom contains such a table, which identifies the chunk for each sample in a media. Each entry in the table contains a first chunk field, a samples per chunk field, and a sample description ID field. From this information, you can ascertain where samples reside in the media data.

Figure 4-31 The layout of a sample-to-chunk table

| Sample-to-chunk table | | | Fields |
|-----------------------|-------------------|--------------------|--------|
| First chunk | Samples per chunk | Sample description | |
| 4 | 4 | 4 | Bytes |

You define a sample-to-chunk table by specifying these elements:

- First chunk. The first chunk number using this table entry.
- Samples per chunk. The number of samples in each chunk.
- Sample description ID. The identification number associated with the sample description containing the sample. For details on sample description atoms, see “Sample Description Atoms,” which begins on page 4-36.

Figure 4-32 shows an example of a sample-to-chunk table that is based on the data stream shown in Figure 4-22.

Figure 4-32 An example of a sample-to-chunk table

| Sample-to-chunk table | | | |
|-----------------------|----|----|-----------------------|
| 1 | 3 | 5 | First chunk |
| 3 | 1 | 1 | Samples per chunk |
| 23 | 23 | 24 | Sample description ID |

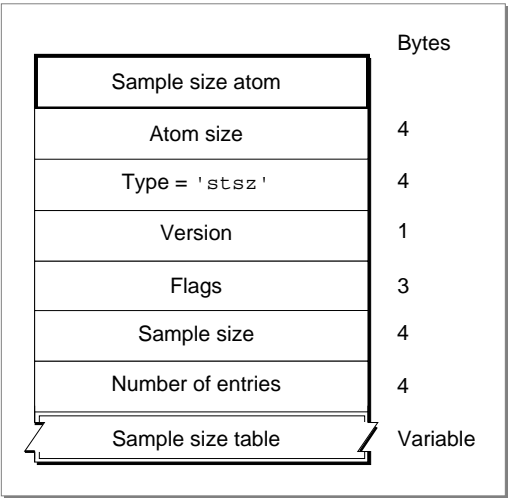
Movie Resource Formats

Each table entry corresponds to a set of consecutive chunks, each of which contains the same number of samples. Furthermore, each of the samples in these chunks must use the same sample description (see “Sample Description Atoms,” which begins on page 4-36). Whenever the number of samples per chunk or the sample description changes, you must create a new table entry. If all the chunks have the same number of samples per chunk and use the same sample description, this table has one entry.

Sample Size Atoms

You use sample size atoms to identify the size of each sample in the media. Sample size atoms have an atom type of 'stsz'. The sample size atom (shown in Figure 4-33) contains sample size information.

Figure 4-33 The layout of a sample size atom



You define a sample size atom by specifying these elements:

- Size. A long integer that specifies the number of bytes in this sample size atom.
- Type. A long integer that specifies the type (of atom type 'stsz') of the data in this sample size atom.
- Version. A 1-byte specification of the version number of this sample size atom.
- Flags. Three bytes of space for future flags associated with the data in this sample size atom.
- Sample size. The number of bytes in the samples in the sample size table field. If all the samples are the same size, the sample size field of this atom indicates the size of all the samples. If this field is set to 0, then the samples have different sizes, and those sizes are stored in the sample size table.

Movie Resource Formats

- Number of entries. The number of entries in the sample size table contained in the sample size table field of this atom.
- Sample size table. The sample size table, which contains the sample size information. A sample size table contains an entry for every sample. Each table entry contains a size field. There is one table entry for each sample in the media. The table is indexed by sample number—the first entry corresponds to the first sample, the second entry is for the second sample, and so on. The size field contains the size, in bytes, of the sample in question.

Figure 4-34 shows the sample size table for the data stream represented in Figure 4-22 on page 4-34.

Figure 4-34 An example of a sample size table

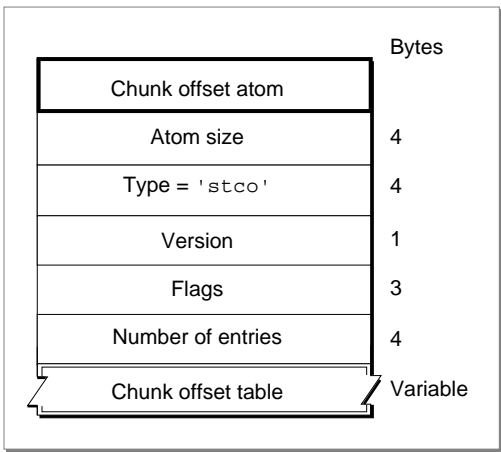
| | |
|------|----------|
| Size | Sample 1 |
| Size | Sample 2 |
| Size | Sample 3 |
| Size | Sample 4 |
| Size | Sample 5 |

Chunk Offset Atoms

Chunk offset atoms identify the location of each chunk of data in the media’s data stream.

Chunk offset atoms have an atom type of 'stco'. The chunk offset atom (shown in Figure 4-35) contains a table of offset information.

Figure 4-35 The layout of a chunk offset atom



You define a chunk offset atom by specifying these elements:

- **Size.** A long integer that specifies the number of bytes in this chunk offset atom.
- **Type.** A long integer that specifies the type of the data in this chunk offset atom (defined by the atom type 'stco').
- **Version.** A 1-byte specification of the version of this chunk offset atom.
- **Flags.** A 3-byte space for future flags associated with this chunk offset atom.
- **Number of entries.** A long integer that specifies the number of entries in the chunk offset table.
- **Chunk offset table.** The chunk offset table, which consists of a number of offset fields. Each entry in the chunk offset table contains an offset field. There is one table entry for each chunk in the media. The table is indexed by chunk number—the first table entry corresponds to the first chunk, the second table entry is for the second chunk, and so on. The offset field contains the byte offset from the beginning of the data stream to the chunk.

Movie Resource Formats

Figure 4-36 shows an example of the chunk offset table for the data stream represented by Figure 4-22 on page 4-34.

Figure 4-36 An example of a chunk offset table

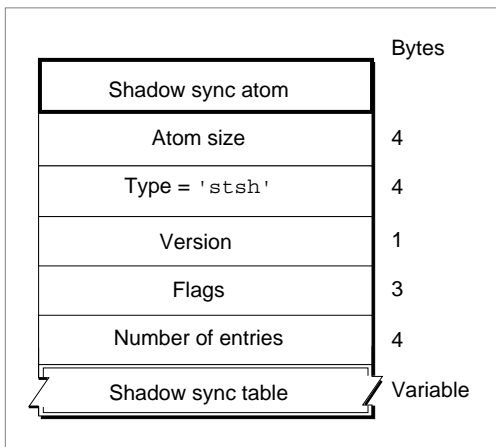
| | |
|--------|---------|
| Offset | Chunk 1 |
| Offset | Chunk 2 |
| Offset | Chunk 3 |
| Offset | Chunk 4 |
| Offset | Chunk 5 |

Shadow Sync Atoms

Shadow sync atoms contain self-contained samples that are alternates for existing frame difference samples. Shadow sync atoms are used to optimize random access operations on a movie. Scrubbing is an example of such a random access operation. These atoms are used to enhance playback performance. See the chapter “Movie Toolbox” in this book for details on the `SetMediaShadowSync` and `GetMediaShadowSync` functions, which allow you to create an association between a frame difference sample and a sync sample.

Figure 4-37 shows the layout of a shadow sync atom. Shadow sync atoms have an atom type of 'stsh'. Each shadow sync atom contains a table with a frame difference number and a sync sample number.

Figure 4-37 The layout of a shadow sync atom



You define a shadow sync atom by specifying these elements:

- Size. A long integer that specifies the number of bytes in this shadow sync atom.
- Type. A long integer that specifies the type (defined by the atom type 'stsh') of the data in this shadow sync atom.
- Version. A 1-byte specification of the version number of this shadow sync atom.
- Flags. Three bytes of space for future flags.
- Number of entries. A long integer that specifies how many entries in the shadow sync table are listed in the shadow sync table field of this atom.
- Shadow sync table. The shadow sync table, which contains the shadow sync information. The shadow sync table is shown in Figure 4-38.

Figure 4-38 The layout of a shadow sync table

| Shadow sync table | |
|--------------------------------|--------------------|
| Frame difference sample number | Sync sample number |
| 4 | 4 |

Fields

Bytes

A shadow sync table contains a frame difference sample number and a sync sample number.

Using Media Information Atoms

This section presents examples using the atoms just described. These examples are intended to help you understand the relationships between these atoms. The first example, “Finding a Sample,” describes the steps that the video media handler uses to find the sample that contains the media data for a particular time in a media. The second example, “Finding a Key Frame,” describes the steps that the video media handler uses to find an appropriate key frame for a specific time in a movie.

Finding a Sample

When it displays a movie or track, QuickTime tells the appropriate media handler to access the media data for a particular time. The media handler must correctly interpret the data stream to retrieve the requested data. In the case of video media, the media handler traverses several atoms to find the location and size of a sample for a given media time. The media handler does the following:

1. Determines the time in the media time coordinate system.
2. Examines the time-to-sample atom to determine the sample number that contains the data for the specified time.
3. Scans the sample-to-chunk atom to discover which chunk contains the sample in question.
4. Extracts the offset to the chunk from the chunk offset atom.
5. Finds the offset within the chunk by using the sample size atom.

Finding a Key Frame

Finding a key frame for a specified time in a movie is slightly more complicated than finding a sample for a specified time. The media handler must use the sync sample atom and the time-to-sample atom together in order to find a key frame. The media handler does the following:

1. Examines the time-to-sample atom to determine the sample number that contains the data for the specified time.
2. Scans the sync sample atom to find the key frame that precedes the sample number chosen in step 1.
3. Scans the sample-to-chunk atom to discover which chunk contains the key frame.
4. Extracts the offset to the chunk from the chunk offset atom.
5. Finds the offset within the chunk by using the sample size atom.

This chapter has described the format of QuickTime movie resources for those developers who need to know about the content of movie resources. The knowledge you have gained about movie resources should help you in the creation of movies on other computers and in the process of importing them to the Macintosh environment, or in the interpretation of QuickTime movies on other types of computers.