QuickTime File Format Specification, May 1996

Developer Press

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

Simultaneously published in the United States and Canada.

LIMITED WARRANTY ON MEDIA AND REPLACEMENT

If you discover physical defects in the manual or in the media on which a software product is distributed, ADC will replace the media or manual at no charge to you provided you return the item to be replaced with proof of purchase to ADC.

ALL IMPLIED WARRANTIES ON THIS MANUAL, INCLUDING IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, ARE LIMITED IN DURATION TO NINETY (90) DAYS FROM THE DATE OF THE ORIGINAL RETAIL PURCHASE OF THIS PRODUCT.

Even though Apple has reviewed this manual, APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS MANUAL, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS MANUAL IS SOLD "AS IS," AND YOU, THE PURCHASER, ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY.

IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT OR INACCURACY IN THIS MANUAL, even if advised of the possibility of such damages.

THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESS OR IMPLIED. No Apple dealer, agent, or employee is authorized to make any modification, extension, or addition to this warranty.

Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.

Contents

Figures and Tables vii

Preface About the QuickTime File Format ix

For More Information ix

Chapter 1 QuickTime File Format 1

Atoms 2
QT Atoms 3
File Format 5
Free Space Atoms 7
Movie Data Atoms 7
Preview Atoms 7
Movie Atoms 8
The Movie Atom 11
Movie Header Atoms 12
Color Table Atoms 14
Track Atoms 15
Track Header Atoms 17
Clipping Atoms 19
Clipping Region Atoms 20
Track Matte Atoms 20
Compressed Matte Atoms 21
Edit Atoms 22
Edit List Atoms 23
Track Load Settings Atoms 24
Track Reference Atoms 26
Track Input Map Atoms 27
Media Atoms 30
Media Header Atoms 31
Handler Reference Atoms 33
Media Information Atoms 34
Video Media Information Atoms 34
Video Media Information Header Atoms 35
Sound Media Information Atoms 37
Sound Media Information Header Atoms 37
Base Media Information Atoms 38
Base Media Information Header Atoms 39
Base Media Info Atoms 39
Data Information Atoms 41

Data Reference Atoms 43
Sample Atoms 44
Sample Table Atoms 45
Sample Description Atoms 47
Time-to-Sample Atoms 48
Sync Sample Atoms 50
Sample-to-Chunk Atoms 51
Sample Size Atoms 53
Chunk Offset Atoms 55
Using Sample Atoms 56
Finding a Sample 56
Finding a Key Frame 57
User Data Atoms 57
Media Data Atom Types 59
Video Media 59
Video Sample Description 59
Video Sample Data 61
Sound Media 64
Sound Sample Description 64
Sound Sample Data 65
Time Code Media 65
Time Code Sample Description 66
Time Code Media Information Atom 66
Time Code Sample Data 67
Text Media 68
Text Sample Description 68
Text Sample Data 69
Music Media 70
Music Sample Description 70
Music Sample Data 71
MPEG Media 71
MPEG Sample Description 71
MPEG Sample Data 71
Sprite Media 71
Sprite Sample Description 71
Sprite Sample Data 72
Base Media 73
Base Sample Description 73
Base Sample Data 73
Tween Media 73
Tween Sample Description 73
Tween Sample Data 73
3D Media 74
3D Sample Description 74
3D Sample Data 75
Basic Data Types 75
Language Code Values 75

Calendar Date and Time Values 77
Matrices 77
Graphics Modes 78
RGB Colors 78
Examples 78
Creating Video Tracks at 30 Frames-per-second 79
Creating Video Tracks at 29.97 Frames-per-second 79
Creating Audio Tracks at 44.1Khz 80
Creating a Time Code Track for 29.97 FPS Video 80
Playing With Edit Lists 84
Interleaving Movie Data 85
Referencing Two Data Files With a Single Track 86

Index IN-1

Figures and Tables

Chapter 1 QuickTime File Format 1

Figure 1-1 A sample QuickTime atom 3
Figure 1-2 QT atom layout 4
Figure 1-3 The structure of a QuickTime file 6
Figure 1-4 The layout of a preview atom 8
Figure 1-5 Sample organization of a one-track video movie atom 10
Figure 1-6 The layout of a movie atom 11
Figure 1-7 The layout of a movie header atom 13
Figure 1-8 The layout of a color table atom 15
Figure 1-9 The layout of a track atom 16
Figure 1-10 The layout of a track header atom 17
Figure 1-11 The layout of a clipping atom 19
Figure 1-12 The layout of a track matte atom 21
Figure 1-13 The layout of an edit atom 23
Figure 1-14 The layout of an edit list table 24
Figure 1-15 The layout of a track load settings atom 25
Figure 1-16 The layout of a track reference atom 26
Figure 1-17 The layout of a track input map atom 28
Figure 1-18 The layout of a media atom 31
Figure 1-19 The layout of a media header atom 32
Figure 1-20 The layout of a handler reference atom 33
Figure 1-21 The layout of a media information atom for video 35
Figure 1-22 The layout of a media information header atom for video 36
Figure 1-23 The layout of a media information atom for sound 37
Figure 1-24 The layout of a sound media information header atom 38
Figure 1-25 The layout of the base media information atom 39
Figure 1-26 The layout of the base media info atom 40
Figure 1-27 The layout of a data information atom 42
Figure 1-28 Samples in a media 45
Figure 1-29 The layout of a sample table atom 46
Figure 1-30 The layout of a sample description atom 47
Figure 1-31 The layout of a time-to-sample atom 48
Figure 1-32 The layout of a time-to-sample table 49
Figure 1-33 An example of a time-to-sample table 50
Figure 1-34 The layout of a sync sample atom 50
Figure 1-35 The layout of a sync sample table 51
Figure 1-36 The layout of a sample-to-chunk atom 52
Figure 1-37 The layout of a sample-to-chunk table 52
Figure 1-38 An example of a sample-to-chunk table 53
Figure 1-39 The layout of a sample size atom 54
Figure 1-40 An example of a sample size table 55
Figure 1-41 The layout of a chunk offset atom 55
Figure 1-42 An example of a chunk offset table 56
Figure 1-43 The layout of a user data atom 57
Figure 1-44 How display matrices are used in QuickTime 77

Figure 1-45 Noninterleaved movie data 86
Figure 1-46 Interleaved movie data 86

Table 1-1 QuickTime file basic atom types 6
Table 1-2 Track reference types 27
Table 1-3 Input types 29
Table 1-4 Data reference types 44
Table 1-5 User data list entry types 58
Table 1-6 Image compression formats 59
Table 1-7 Video sample description extensions 61
Table 1-8 Sound data format types 64
Table 1-9 Text sample extensions 70
Table 1-10 Sprite properties 72
Table 1-11 Tween type values 74
Table 1-12 QuickTime language code values 75
Table 1-13 QuickTime graphics modes 78

About the QuickTime File Format

This book describes the format and content of QuickTime files. It is intended for developers who need to work with QuickTime files outside the context of the QuickTime environment. For example, if you are developing a non-QuickTime application that imports QuickTime files, you need to understand the material in this book. On the other hand, if you are using QuickTime on any of its supported platforms, you do not necessarily need to be familiar with the file format information presented here.

This book describes QuickTime files in general, rather than how they are supported on a specific computing platform or in a specific programming language. As a result, the file format information is presented in a tabular manner, rather than in coded data structures. Similarly, field names are presented in English rather than as programming language tags. Furthermore, to the extent possible, data types are described generically. For example, this book uses "32-bit signed integer" rather than "`long`" to define a 32-bit integer value. Based on the information provided here, you should be able to create appropriate data structure specifications for your environment.

Finally, QuickTime files are used to store QuickTime movies, as well as other time-based data. If you are writing an application that parses QuickTime files, you should recognize that there may be non-movie data in the files.

For More Information

The *Apple Developer Catalog* (ADC) is Apple Computer's worldwide source for hundreds of development tools, technical resources, training products, and information for anyone interested in developing applications on Apple computer platforms. Customers receive the *Apple Developer Catalog* featuring all current versions of Apple development tools and the most popular third-party development tools. ADC offers convenient payment and shipping options, including site licensing.

To order products or to request a complimentary copy of the *Apple Developer Catalog*, contact

Apple Developer Catalog
Apple Computer, Inc.
P.O. Box 319
Buffalo, NY 14207-0319

Telephone	1-800-282-2732 (United States) 1-800-637-0029 (Canada) 716-871-6555 (International)
Fax	716-871-6511
AppleLink	ORDER.ADC
Internet	order.adc@applelink.apple.com

QuickTime File Format

This document describes how QuickTime movies are stored on disk. The QuickTime file format is designed to accommodate the varied kinds of data that need to be stored in order to work with digital media. Because the file format can be used to describe almost any media structure, it is an ideal format for the exchange of digital media between applications, regardless of the platform on which the application may be running.

This document assumes that the reader is familiar with the basic concepts of digital video and digital audio, as well as with QuickTime. For a complete description of QuickTime concepts, including time coordinate systems and how spatial tracks are composited together, please refer to *Inside Macintosh: QuickTime*.

A QuickTime file stores the description of the media separately from the media data. The description, or meta-data, is called the **movie** and contains information such as the number of tracks, video compression format, and timing information. The movie also contains an index of where all the media data is stored. The media data is all of the actual sample data, such as video frames and audio samples. The media data may be stored in the same file as the QuickTime movie, in a separate file, or in several files.

Before explaining the specifics of how a QuickTime movie is stored, it is important to first understand the basic units that are used to construct QuickTime files. QuickTime uses two basic structures for storing information: **atoms** and **QT atoms**. Both atoms and QT atoms allow you to construct arbitrarily complex hierarchical data structures. Both also allow applications to ignore data they don't understand.

Atom types are specified by a four-character code. Apple Computer reserves all four-character codes consisting entirely of lower case letters.

Unless otherwise stated, all data in a QuickTime movie is stored in big-endian (Motorola) byte ordering.

Finally, all version fields must be set to 0, unless this document states otherwise.

Atoms

The basic data unit in a QuickTime file 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. Both the `size` and `type` fields are 32-bit integers.

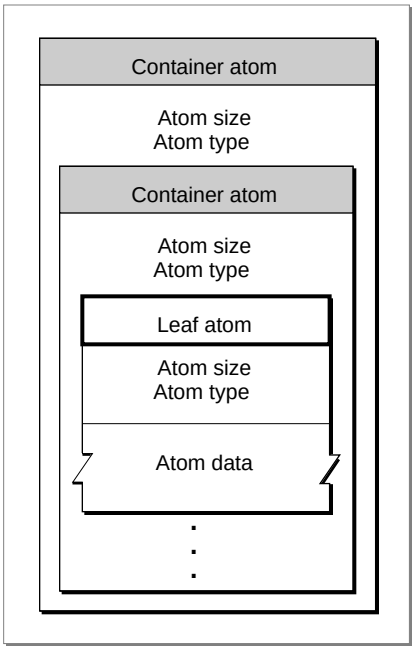
Atoms are recursive in nature. That is, one atom may contain one or more other atoms of varying type. For example, a movie atom contains one track atom for each track in the movie. The track atoms, in turn, contain one or more media atoms each, along with other atoms that define other track and movie characteristics. This hierarchical structure of atoms is referred to as a containment hierarchy.

The format of the data stored within a given atom cannot be determined based only on the `type` field of that atom. That is, an atom's use is determined by its context. A given atom type may have different usages when stored within atoms of different types. This means that all QuickTime file readers must take into consideration not only the atom type, but the atom's containment hierarchy.

Figure 1-1 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.

Atoms within container atoms do not have to be in any particular order, with the exception of handler description atoms. Handler description atoms must come before their data. For example, a media handler description atom must come before a media information atom. A data handler description atom must come before a data information atom.

Figure 1-1 A sample QuickTime atom



Atoms consist of a header, followed by atom data. An atom header consists of the following fields.

Field descriptions

Atom size — A 32-bit integer that indicates the size of the atom, including both the atom header and the atom's contents. If the atom is a leaf atom, then this field contains the size of the single atom. The size of a container atom includes all of its contained atoms.

Type — A 32-bit integer that contains the type of the atom.

The only way to interpret the atom's data is by knowing the type of data that is stored in an atom of a particular type.

QT Atoms

Given the limitations of the structure of the simple atom, Apple has created a new, enhanced data structure called a QT atom. QT atoms provide a more general purpose storage format and remove some of the ambiguities that arise when using simple atoms.

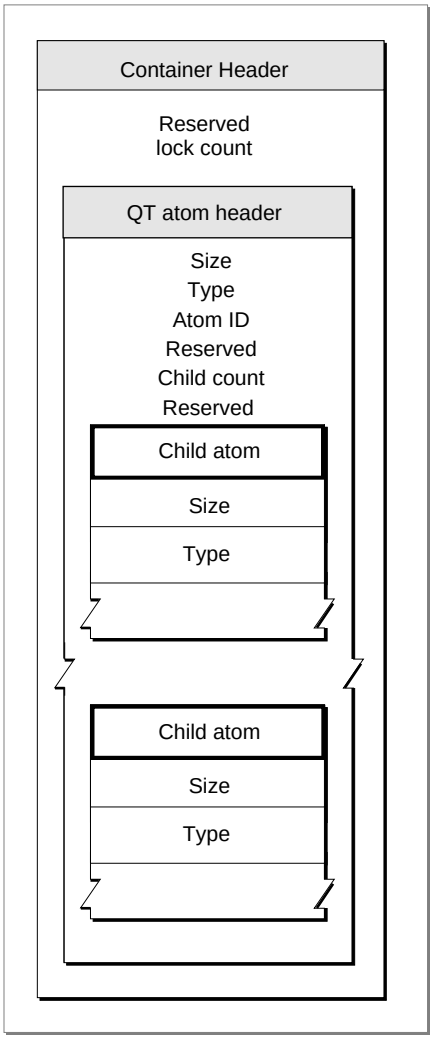
In particular, with simple atoms there is no way to know if an atom is a leaf node or whether it contains other atoms, or both, without specific knowledge about the atom. Using QT atoms, a given node is either is a leaf node or a container node. There is no ambiguity. Furthermore, QT atoms allow for multiple atoms of a given type to be specified through identification numbers. While QT atoms are a more powerful data structure, they require more overhead in the file.

The QuickTime file format uses both atoms and QT atoms. In general, newer parts of the QuickTime file format use QT atoms, and older parts use atoms. When defining new QuickTime structures, you should use QT atoms whenever practical.

Figure 1-2 depicts the layout of a QT atom. Each QT atom starts with a QT atom container header, followed by the root atom. The root atom's type is determined by the QT atom's type. The root atom contains any other atoms that are part of the structure.

Each container atom starts with a QT atom header followed by the atom's contents. The contents are either child atoms or data, but never both. If an atom contains children it also contains all of its children's data and their descendants. The root atom is always present and never has any siblings.

Figure 1-2 QT atom layout



A QT atom container header contains the following data.

Field descriptions

Reserved	A 10-byte element that must be set to 0.
Lock count	A 16-bit integer that must be set to 0.

Each QT atom header contains the following data.

Field descriptions

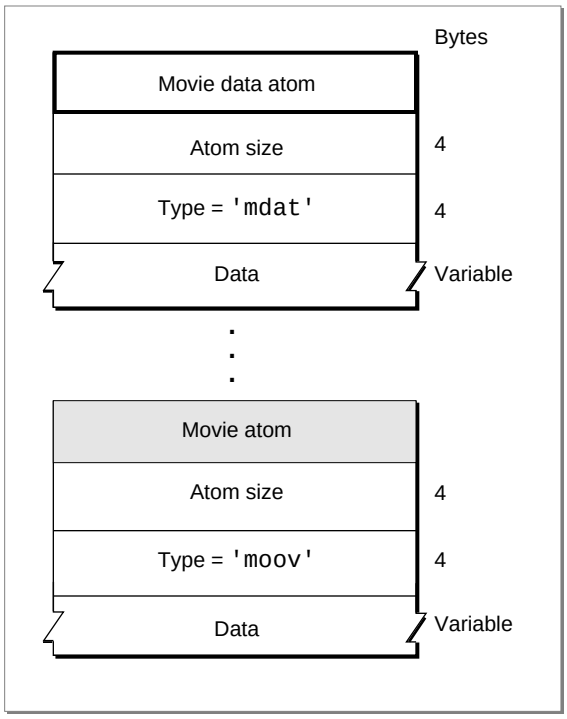
Size	A 32-bit integer that indicates the size of the atom in bytes, including both the QT atom header and the atom's contents. If the atom is a leaf atom, then this field contains the size of the single atom. The size of container atoms includes all of the contained atoms. You can walk the atom tree using the Size and Child count fields (described below).
Type	A 32-bit integer that contains the type of the atom. If this is the root atom, the type value is set to `'sean'`.
Atom ID	A 32-bit integer that contains the atom's ID value. This value must be unique among its siblings. The root atom always has an Atom ID value of 1.
Reserved	A 16-bit integer that must be set to 0.
Child count	A 16-bit integer that specifies the number of child atoms that an atom contains. This count only includes immediate children. If this field is set to 0, the atom is a leaf atom and only contains data.
Reserved	A 32-bit integer that must be set to 0.

File Format

A QuickTime file is simply a collection of atoms. QuickTime does not impose any rules about the order of these atoms.

Figure 1-3 depicts a typical QuickTime file.

Figure 1-3 The structure of a QuickTime file



In file systems that support file name extensions, QuickTime file names typically have an extension of `".mov"`. On the Macintosh platform, QuickTime files have a file type of `"MooV"`. On the Macintosh, the movie atom may be stored as a Macintosh resource using the Resource Manager. The resource has a type of `'moov'`. All media data is stored in the data fork.

As previously discussed, QuickTime files consist of atoms, each with an appropriate atom type. A few of these types are considered basic atom types and form the structure within which the other atoms are stored. Table 1-1 lists the currently supported basic atom types.

Table 1-1 QuickTime file basic atom types

Atom type	Use
`'free'`	Unused space available in file
`'skip'`	Unused space available in file
`'mdat'`	Movie data—usually this data can only be interpreted by using the movie resource
`'pnot'`	Reference to movie preview data
`'moov'`	Movie resource

While it is true that QuickTime imposes no strict order on a movie's atoms, it is often convenient if the movie atom appears near the front of the file. For example, an application that plays a movie over a network would not necessarily have access to the entire movie at all times. If the movie atom is stored at the beginning of the file, the application can use the meta-data to understand the movie's content as it is acquired over the network.

The following sections describe each of these basic atom types in more detail, including descriptions of the atoms that each basic atom may contain.

Free Space Atoms

Both free and skip atoms designate unused space in the movie data file. These atoms consist only of an atom header (atom size and type fields), followed by the appropriate number of bytes of free space. When reading a QuickTime movie, your application may safely skip these atoms. When writing or updating a movie, you may re-use the space associated with these atom types.

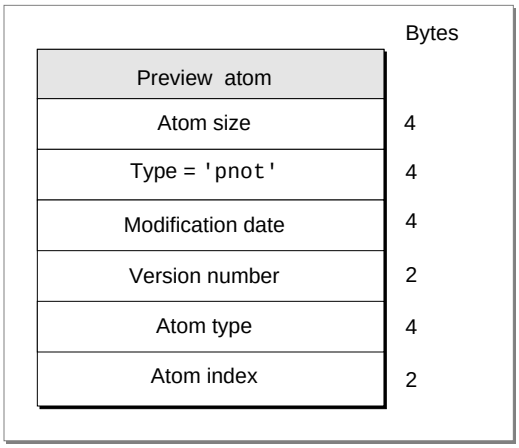
Movie Data Atoms

As with the free and skip atoms, the movie data atom is structured quite simply. It consists of an atom header (atom size and type fields), followed by the movie's media data. Your application can understand the data in this atom only by using the meta-data stored in the movie atom.

Preview Atoms

The preview atom contains information that allows you to find the preview image associated with a QuickTime movie. The preview image, or poster, is a representative image suitable for display to the user in, say, file-open dialogs. Figure 1-4 depicts the layout of a preview atom.

Figure 1-4 The layout of a preview atom



The preview atom has an atom type value of `'pnot'` and, following its atom header, contains the following fields.

Field descriptions

Size	A 32-bit integer that specifies the number of bytes in this preview atom.
Type	A 32-bit integer that identifies the atom type; this field must be set to `'pnot'`.
Modification date	A 32-bit unsigned integer containing a date that indicates when the preview was last updated. The date is in standard Macintosh format.
Version number	A 16-bit integer that must be set to 0.
Atom type	A 32-bit integer that indicates the type of atom that contains the preview data. Typically, this is set to `'PICT'` to indicate a QuickDraw picture.
Atom index	A 16-bit integer that identifies which atom of the specified type is to be used as the preview. Typically, this field is set to 1 to indicate that you should use the first atom of the type specified in the Atom type field.

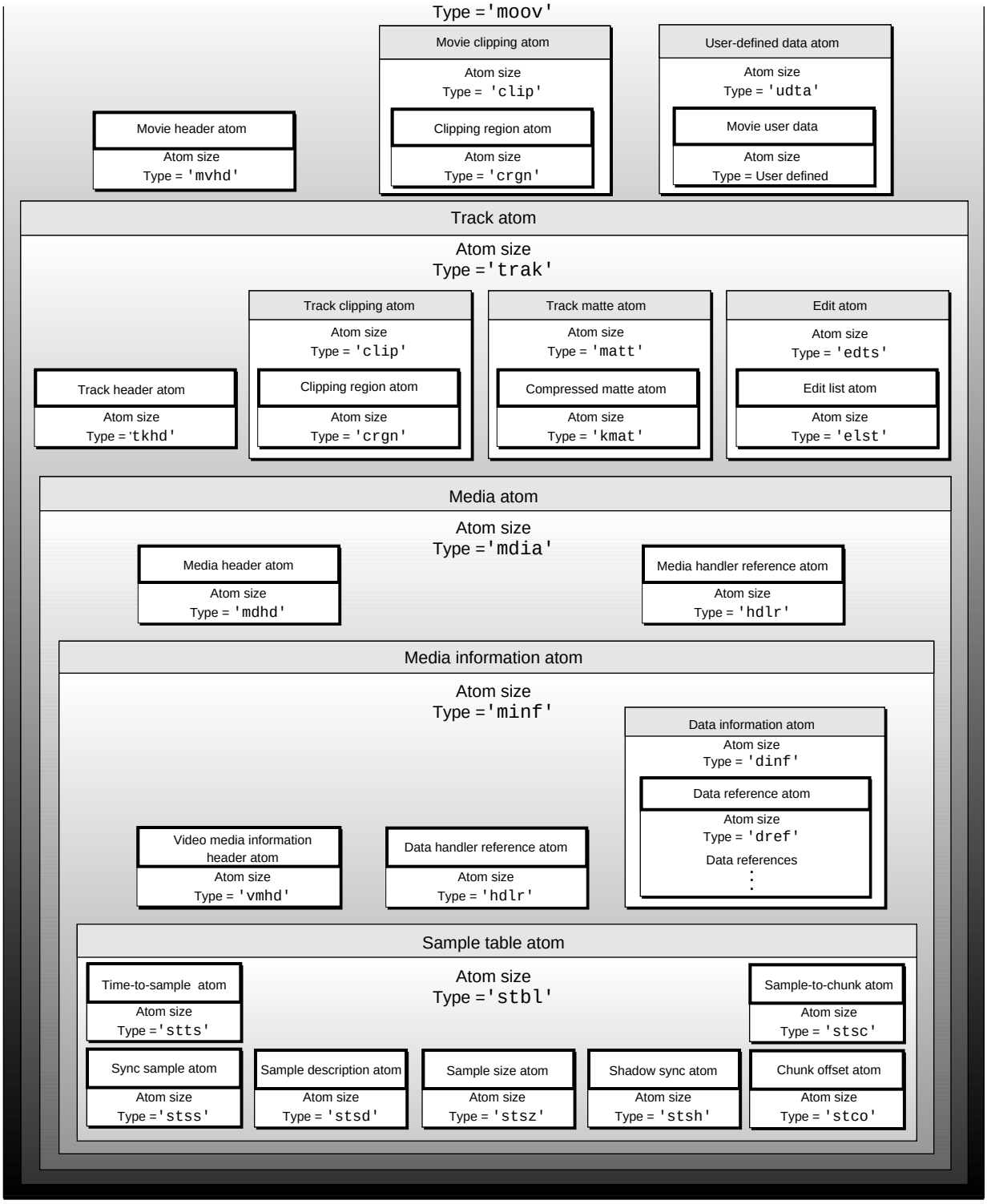
Movie Atoms

Movie atoms have an atom type of `'moov'`. These atoms act as a container for the information that describes a movie's data. This information, or meta-data, is stored in a number of different types of atoms. As such, the movie atom is essentially a container of other atoms. At the highest level, movie atoms contain track atoms, which in turn contain media atoms. At the lowest level you find the leaf atoms, which contain the actual data, usually in the form of a table or a data stream.

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. Both of these atoms are discussed later in this book.

Figure 1-5 provides a conceptual view of the organization of a simple, one-track QuickTime movie. Each nested box in the illustration represents an atom that belongs to its containing atom. The figure does not show the data regions of any of the atoms. These areas are described in the sections that follow.

Figure 1-5 Sample organization of a one-track video movie atom



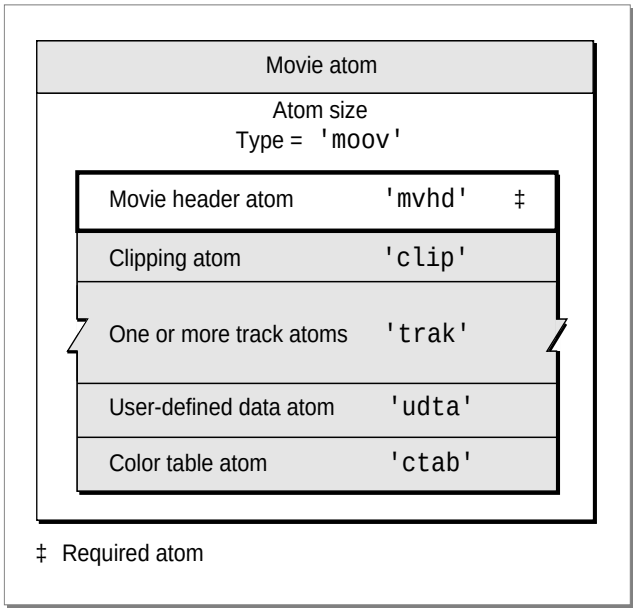
The Movie Atom

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.

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'`), a color table atom (`'ctab'`), and user data (`'udta'`).

Figure 1-6 shows the layout of a movie atom. The movie header atom is the only required atom in the movie atom.

Figure 1-6 The layout of a movie atom



A movie atom contains the following information.

Field descriptions

Size — The number of bytes in this movie atom.

Type — The type of this movie atom; this field must be set to `'moov'`.

Movie header — The movie header atom associated with this movie. See the next section for details on the movie header atom.

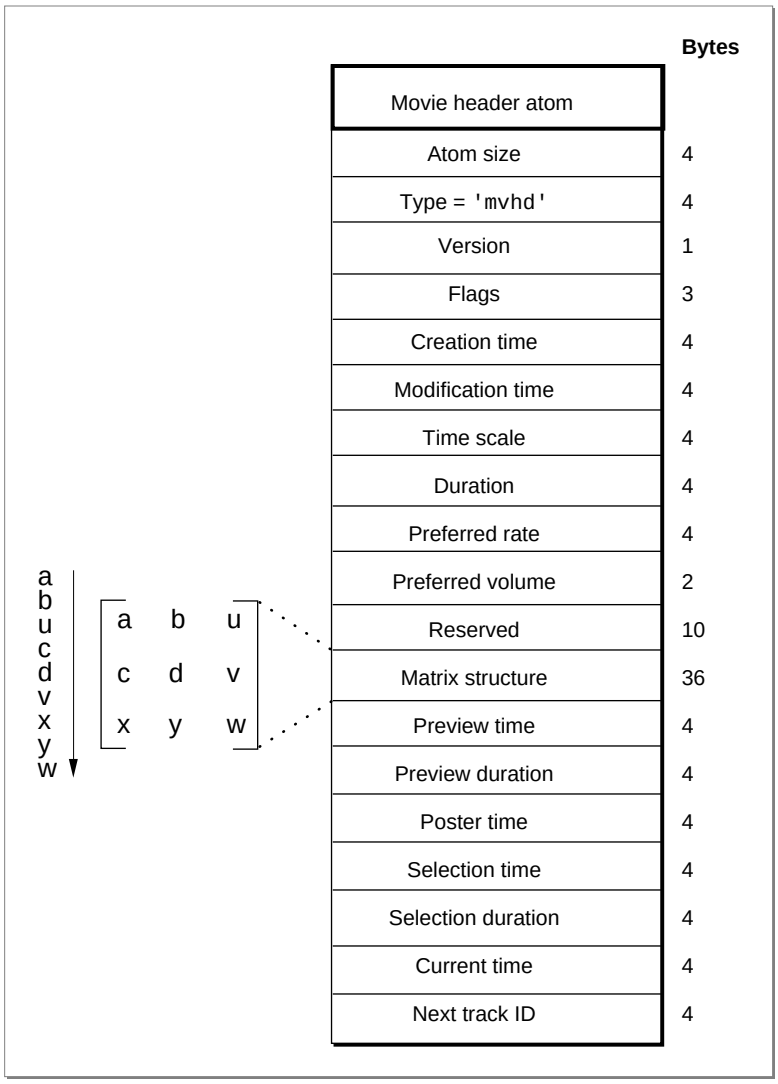
Movie clipping atom — The clipping atom associated with this movie. See “Clipping Atoms,” beginning on page 19, for more information.

Track list	One or more track atoms associated with this movie. See “Track Atoms,” beginning on page 15, for details on track atoms and their associated atoms.
User data	The user data atom associated with this movie. See “User Data Atoms” on page 57 for more information about user data atoms.
Color table	The color table atom associated with this movie. See “Color Table Atoms” on page 14 for a discussion of the color table atom.

Movie Header Atoms

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. It has an atom type value of `'mvhd'`. Figure 1-7 shows the layout of a movie header atom. The movie header atom is a leaf atom.

Figure 1-7 The layout of a movie header atom



You define a movie header atom by specifying the following data elements.

Field descriptions

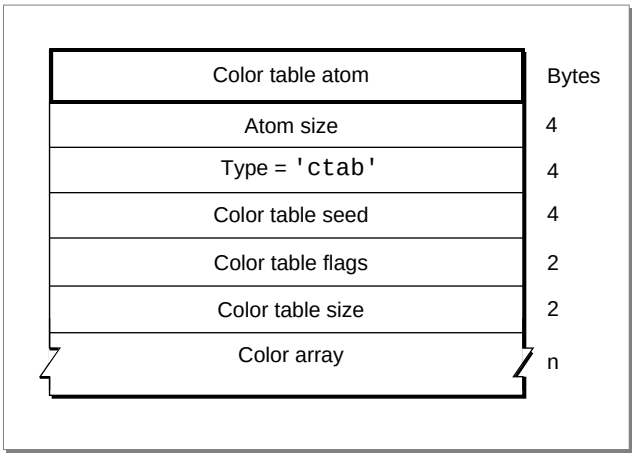
Size	A 32-bit integer that specifies the number of bytes in this movie header atom.
Type	A 32-bit integer that identifies the atom type; this field must be set to `'mvhd'`.
Version	A 1-byte specification of the version of this movie header atom.
Flags	Three bytes of space for future movie header flags.
Creation time	A 32-bit integer that specifies (in seconds since midnight, January 1, 1904) when the movie atom was created.

Modification time	A 32-bit 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. Note that this property is derived from the movie's tracks. The value of this field corresponds to the duration of the longest track in the movie.
Preferred rate	A 32-bit fixed-point number that specifies the rate at which to play this movie. A value of 1.0 indicates normal rate.
Preferred volume	A 16-bit fixed-point number that specifies how loud to play this movie's sound. A value of 1.0 indicates full volume.
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.
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.
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 32-bit integer that indicates a value to use for the track ID number of the next track added to this movie. Note that 0 is not a valid track ID value.

Color Table Atoms

Color table atoms define a list of preferred colors for displaying the movie on devices that only support 256 colors. The list may contain up to 256 colors. These optional atoms have a type value of `'ctab'`. The color table atom contains a Macintosh Color Table data structure. Figure 1-8 shows the layout of a color table atom.

Figure 1-8 The layout of a color table atom



The color table atom contains the following data elements.

Field descriptions

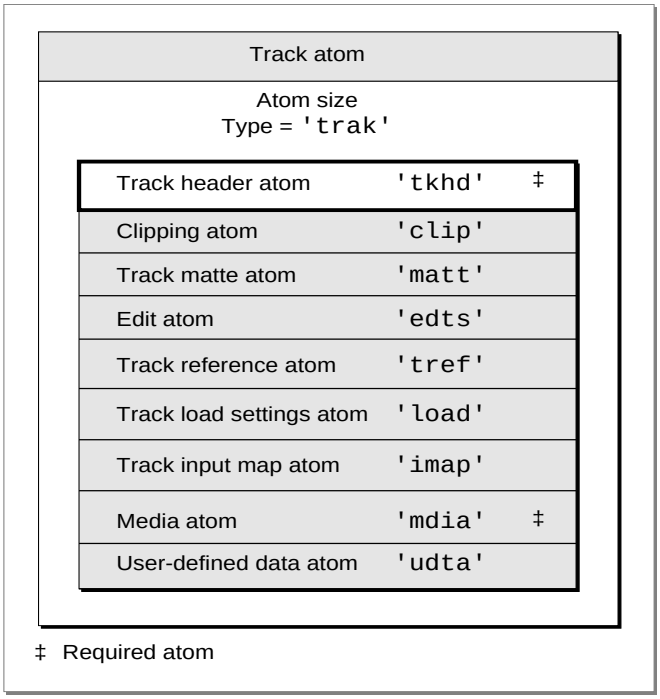
Size	A 32-bit integer that specifies the number of bytes in this color table atom.
Type	A 32-bit integer that identifies the atom type; this field must be set to `'ctab'`.
Color table seed	A 32-bit integer that must be set to 0.
Color table flags	A 16-bit integer that must be set to 0x8000
Color table size	A 16-bit integer that indicates the number of colors in the following color array. This is a zero-relative value; setting this field to 0 means that there is one color in the array.
Color array	An array of colors. Each color is made up of four unsigned 16-bit integers. The first integer must be set to 0, the second is the red value, the third is the green value, and the fourth is the blue value.

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 1-9 shows the layout of a track atom. Track atoms have an atom type value of `'trak'`. The track atom requires the track header atom (`'tkhd'`) and the media atom (`'mdia'`). Other child atoms are optional and may include a track clipping atom (`'clip'`), a track matte atom (`'matt'`), an edit atom (`'edts'`), a track reference atom (`'tref'`), a track input map atom (`'imap'`), and a user data atom (`'udta'`).

Figure 1-9 The layout of a track atom



Track atoms contain the following data elements.

Field descriptions

Size	A 32-bit integer that specifies the number of bytes in this track atom.
Type	A 32-bit integer that identifies the atom type; this field must be set to `'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,” beginning on page 19, for more information.
Track matte	The track matte atom associated with this track. See “Track Matte Atoms,” beginning on page 20, for more information.
Edits	The edit atom associated with this track. See “Edit Atoms,” beginning on page 22, for details.
Track references	The track reference atom associated with this track. See “Track Reference Atoms,” beginning on page 26, for details.
Track load settings	The track load settings atom associated with this track. See “Track Load Settings Atoms,” beginning on page 24, for details.
Track input map	The track input map atom associated with this track. See “Track Input Map Atoms,” beginning on page 27, for details.
Media	The media atom associated with this track. See “Media Atoms,” beginning on page 30, for details.

User data — The user data atom associated with this track. See "User Data Atoms" on page 57 for more information.

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 1-10 shows the structure of a track header atom.

Figure 1-10 The layout of a track header atom

Track header atom	Bytes
Atom size	4
Type = `'tkhd'`	4
Version	1
Flags	3
Creation time	4
Modification time	4
Track ID	4
Reserved	4
Duration	4
Reserved	8
Layer	2
Alternate group	2
Volume	2
Reserved	2
Matrix structure	36
Track width	4
Track height	4

The track header atom contains the track characteristics for the track, including temporal, spatial, and volume information.

Track header atoms contain the following data elements.

Field descriptions

Size — A 32-bit integer that specifies the number of bytes in this track header atom.

Type — A 32-bit integer that identifies the atom type; this field must be set to `'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. These flags indicate how the track is used in the movie. The following flags are valid (all flags are enabled when set to 1).

Track enabled
: Indicates that the track is enabled. Flag value is 0x0001.

Track in movie
: Indicates that the track is used in the movie. Flag value is 0x0002.

Track in preview
: Indicates that the track is used in the movie's preview. Flag value is 0x0004.

Track in poster
: Indicates that the track is used in the movie's poster. Flag value is 0x0008.

Creation time — A 32-bit integer that indicates (in seconds since midnight, January 1, 1904) when the track header was created.

Modification time — A 32-bit integer that indicates (in seconds since midnight, January 1, 1904) when the track header was changed.

Track ID — A 32-bit integer that uniquely identifies the track. A value of 0 must never be used for a track ID.

Reserved — A 32-bit integer that is reserved for use by Apple. Set this field to 0.

Duration — A time value that indicates the duration of this track (in the movie's time coordinate system). Note that this property is derived from the track's edits. The value of this field is equal to the sum of the durations of all of the track's edits.

Reserved — An 8-byte value that is reserved for use by Apple. Set this field to 0.

Layer — A 16-bit integer that indicates this track's spatial priority in its movie. The QuickTime Movie Toolbox uses this value to determine how tracks overlay one another. Tracks with lower layer values are displayed in front of tracks with higher layer values.

Alternate group — A 16-bit 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 16-bit fixed-point value that indicates how loudly this track's sound is to be played. A value of 1.0 indicates normal volume.

Reserved — A 16-bit integer that is reserved for use by Apple. Set this field to 0.

Matrix — The matrix structure associated with this track. See Figure 1-44 on page 77 for an illustration of a matrix structure.

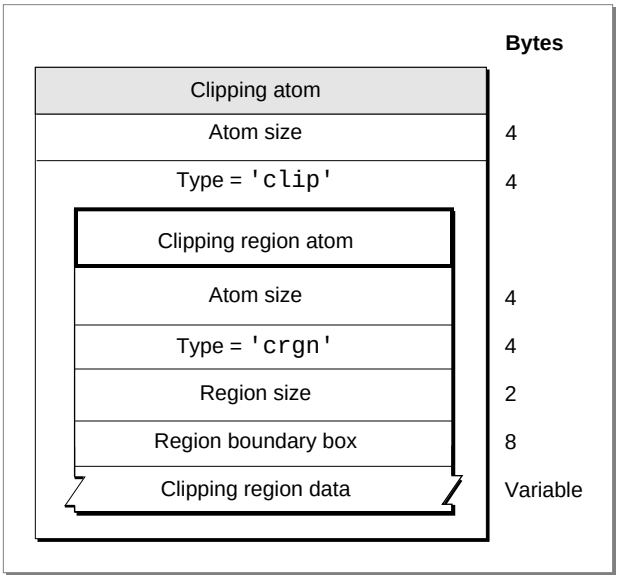
Track width	A 32-bit fixed-point number that specifies the width of this track in pixels.
Track height	A 32-bit fixed-point number that indicates the height of this track in pixels.

Clipping Atoms

Clipping atoms specify the clipping regions for movies and for tracks. The clipping atom has an atom type value of `'clip'`.

Figure 1-11 shows the layout of a clipping atom.

Figure 1-11 The layout of a clipping atom



Clipping atoms contain the following data elements.

Field descriptions

Size	A 32-bit integer that specifies the number of bytes in this clipping atom.
Type	A 32-bit integer that identifies the atom type; this field must be set to `'clip'`.
Clipping region atom	The clipping region atom; these atoms are described in the next section.

Clipping Region Atoms

The clipping region atom contains the data that specifies the clipping region, including its size and bounding box and region. Clipping region atoms have an atom type value of `'crgn'`.

The layout of the clipping region atom is shown in Figure 1-11.

Clipping region atoms contain the following data elements.

Field descriptions

Size — A 32-bit integer that specifies the number of bytes in this clipping region atom.

Type — A 32-bit integer that identifies the atom type; this field must be set to `'crgn'`.

Region size

Region boundary box

Clipping region data

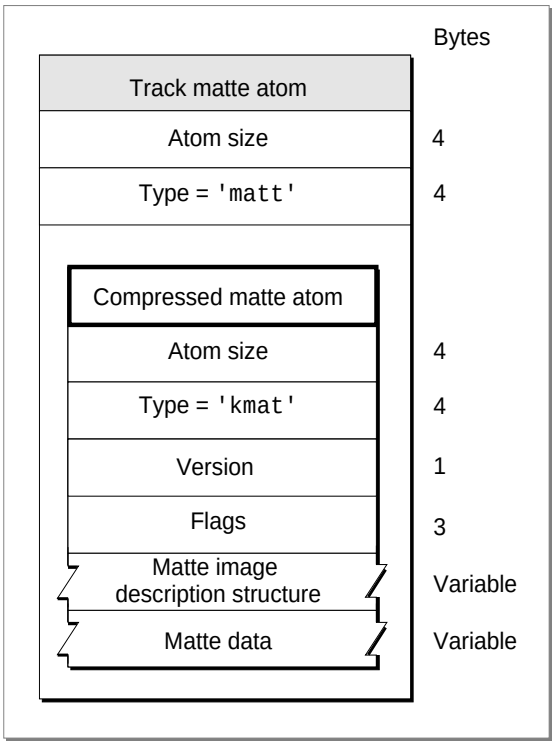
The Region size, Region boundary box, and Clipping region data fields constitute a QuickDraw region. See *Inside Macintosh: Imaging* for details on QuickDraw regions.

Track Matte Atoms

Track matte atoms are used to visually blend the track's image when it is displayed. For a complete description on the use of mattes in QuickTime, see *Inside Macintosh: QuickTime*. Track matte atoms have an atom type value of `'matt'`.

Figure 1-12 shows the layout of a track matte atom.

Figure 1-12 The layout of a track matte atom



Track matte atoms contain the following data elements.

Field descriptions

Size — A 32-bit integer that specifies the number of bytes in this track matte atom.

Type — A 32-bit integer that identifies the atom type; this field must be set to `'matt'`.

Compressed matte atom — The actual matte data in a compressed matte atom. These atoms are described in the next section.

Compressed Matte Atoms

The compressed matte atom specifies the image description structure associated with a particular matte atom. Compressed matte atoms have an atom type value of `'kmat'`.

The layout of the compressed matte atom is shown in Figure 1-12.

Compressed matte atoms contain the following data elements.

Field descriptions

Size — A 32-bit integer that specifies the number of bytes in this compressed matte atom.

Type	A 32-bit integer that identifies the atom type; this field must be set to `'kmat'`.
Version	A 1-byte specification of the version of this compressed matte atom.
Flags	Three bytes of space for flags. Set this field to 0.
Matte image description	An image description structure associated with this matte data. The image description contains detailed information that governs how the matte data is used. See "Video Sample Description" on page 59 for more information about image descriptions.
Matte data	The compressed matte data that is of variable length.

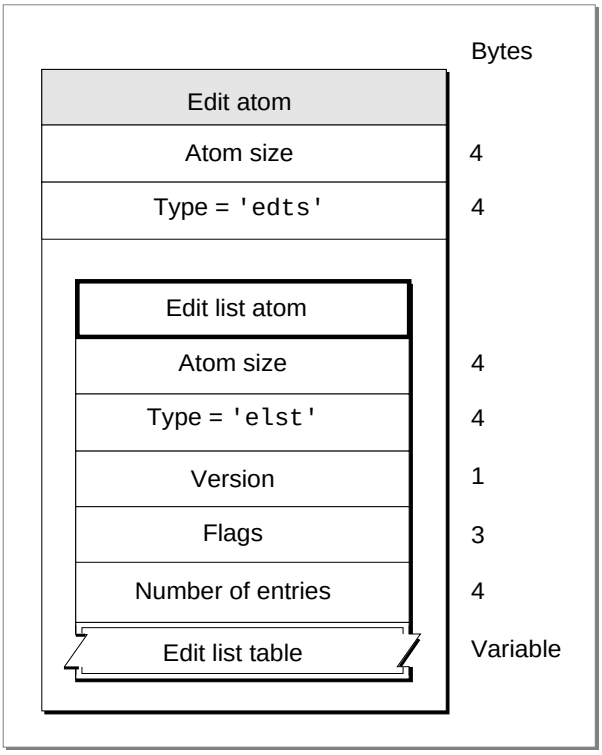
Edit Atoms

You use edit atoms to define the portions of the media that are to be used to build up a track for a movie. The edits themselves are contained in an edit list table, that consists of time offset and duration values for each segment. Edit atoms have an atom type value of `'edts'`.

Figure 1-13 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 used by the track. ◆

Figure 1-13 The layout of an edit atom



Edit atoms contain the following data elements.

Field descriptions

Size	A 32-bit integer that specifies the number of bytes in this edit atom.
Type	A 32-bit integer that identifies the atom type; this field must be set to `'edts'`.
Edit list	The edit list atom that contains the edit list information; these atoms are described in the next section.

Edit List Atoms

You use the edit list atom, shown in Figure 1-13, 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 1-14. Edit list atoms have an atom type value of `'elst'`.

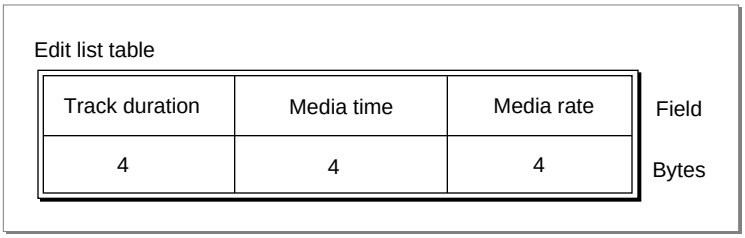
Edit list atoms contain the following data elements.

Field descriptions

Size	A 32-bit integer that specifies the number of bytes in this edit list atom.
Type	A 32-bit integer that identifies the atom type; this field must be set to `'elst'`.

Version	A 1-byte specification of the version of this edit list atom.
Flags	Three bytes of space for flags. Set this field to 0.
Number of entries	A 32-bit integer that specifies the number of entries in the edit list atom that follows.
Edit list table	An array of 32-bit values grouped into entries containing 3 values each.

Figure 1-14 The layout of an edit list table



An edit list table contains the following elements.

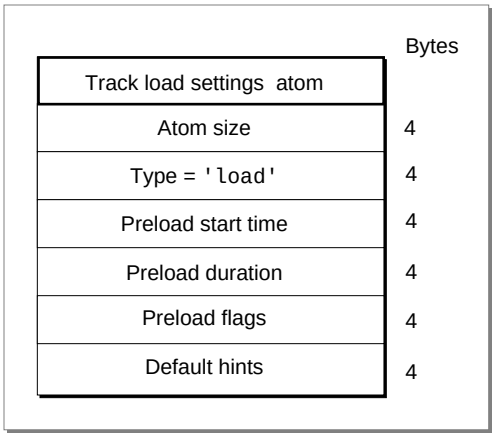
Field descriptions

Track duration	A 32-bit integer that specifies the duration of this edit segment in units of the movie's time scale.
Media time	A 32-bit integer containing the starting time within the media of this edit segment (in media time scale units). If this field is set to –1, it is an empty edit. The last edit in a track should never be an empty edit. Any difference between the movie's duration and the track's duration is expressed as an implicit empty edit.
Media rate	A 32-bit fixed-point number that specifies the relative rate at which to play the media corresponding to this edit segment. This rate value cannot be 0 or negative.

Track Load Settings Atoms

Track load settings atoms contain information that indicates how the track is to be used in its movie. Applications that read QuickTime files can use this information to process the movie data more efficiently. Track load settings atoms are optional and have an atom type value of `'load'`.

Figure 1-15 The layout of a track load settings atom



Track load settings atoms contain the following data elements.

Field descriptions

Size — A 32-bit integer that specifies the number of bytes in this track load settings atom.

Type — A 32-bit integer that identifies the atom type; this field must be set to `'load'`.

Preload start time — A 32-bit integer specifying the starting time, in the movie's time coordinate system, of a segment of the track that is to be preloaded. Used in conjunction with the Preload duration value.

Preload duration — A 32-bit integer specifying the duration, in the movie's time coordinate system, of a segment of the track that is to be preloaded. If the duration is set to –1, it means that the preload segment extends from the preload start time to the end of the track. All media data in the segment of the track defined by the preload start time and preload duration values should be loaded into memory when the movie is to be played.

Preload flags — A 32-bit integer containing flags governing the preload operation. Only two flags are defined and they are mutually exclusive. If Preload flags is set to 1, the track is to be preloaded regardless of whether it is enabled. If Preload flags is set to 2, the track is only preloaded if it is enabled.

Default hints — A 32-bit integer containing playback hints. More than one flag may be enabled. Flags are enabled by setting them to 1. The following flags are defined.

Double buffer — This flag indicates that the track should be played using double-buffered I/O. This flag's value is 0x0020.

High quality — This flag indicates that the track should be displayed at highest possible quality, without regard to real time performance considerations. This flag's value is 0x0100.

Track Reference Atoms

Track reference atoms define relationships between tracks. Track reference atoms allow tracks to specify their relationships to other tracks. For example, if a movie has three video tracks and three sound tracks, track references allow you to identify the related sound and video tracks. Track reference atoms have an atom type value of `'tref'`.

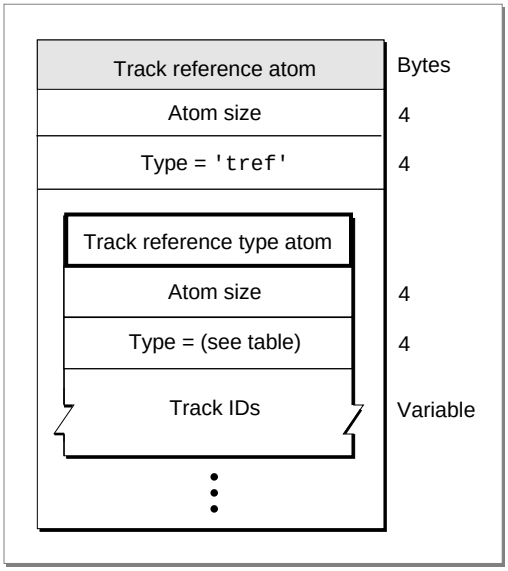
Track references are uni-directional and point from the recipient track to the source track. For example, a video track may reference a time code track to indicate where its time code is stored, but the time code track would not reference the video track. The time code track is the source of time information for the video track.

A single track may reference multiple tracks. For example, a video track could reference a sound track to indicate that the two are synchronized and a time code track to indicate where its time code is stored.

A single track may also be referenced by multiple tracks. For example, both a sound and video track could reference the same time code track if they share the same timing information.

Figure 1-16 shows the layout of a track reference atom.

Figure 1-16 The layout of a track reference atom



A track reference atom contains the following data elements.

Field descriptions

Size	A 32-bit integer that specifies the number of bytes in this track reference atom.
Type	A 32-bit integer that identifies the atom type; this field must be set to `'tref'`.

Reference atoms — A list of track reference type atoms containing the track reference information. These atoms are described next.

Each track reference atom defines relationships with tracks of a specific type. The reference type implies a track type. Table 1-2 shows the track reference types and their descriptions.

Table 1-2 Track reference types

Reference type	Description
`'tmcd'`	Time code. Usually references a time code track.
`'chap'`	Chapter or scene list. Usually references a text track.
`'sync'`	Synchronization. Usually between a video and sound track. Indicates that the two tracks are synchronized. The reference can be from either track to the other, or there may be two references.
`'scpt'`	Transcript. Usually references a text track.
`'ssrc'`	Non-primary source. Indicates that the referenced track should send its data to this track, rather than presenting it. The referencing track will use the data to modify how it presents its data. See the next section, "Track Input Map Atoms," for more information.

Each track reference type atom contains the following data elements.

Field descriptions

Size — A 32-bit integer that specifies the number of bytes in this track reference type atom.

Type — A 32-bit integer that identifies the atom type; this field must be set to one of the values shown in Table 1-2.

Track IDs — A list of track ID values specifying the related tracks. Note that this is one case where track ID values may be set to 0. Unused entries in the atom may have a track ID value of 0. Setting the track ID to 0 may be more convenient that deleting the reference.

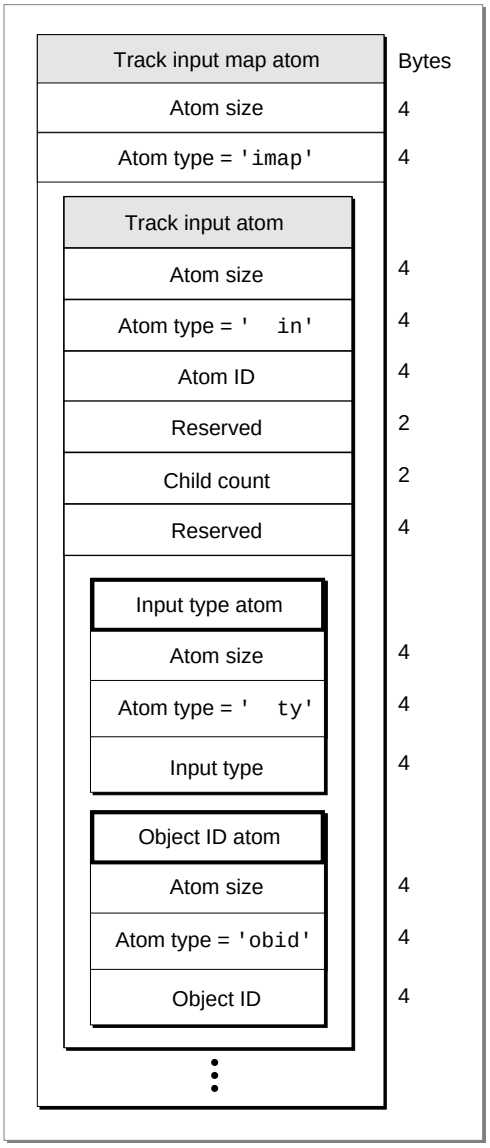
You can determine the number of track references stored in a track reference type atom by subtracting its header size from its overall size and then dividing by the size, in bytes, of a track ID.

Track Input Map Atoms

Track input map atoms define how data being sent to this track from its non-primary sources is to be interpreted. Track references of type `'ssrc'` define a track's secondary data sources. These sources provide additional data that is used when processing the track. Track input map atoms are optional and have an atom type value of `'imap'`.

Figure 1-17 shows the layout of a track input map atom. This atom contains one or more track input atoms. Note that the track input map atom is a QT atom structure.

Figure 1-17 The layout of a track input map atom



Each track input map atom contains the following data elements.

Field descriptions

Size	A 32-bit integer that specifies the number of bytes in this track input map atom.
Type	A 32-bit integer that identifies the atom type; this field must be set to `'imap'`.
Track input atoms	A list of track input atoms specifying how to use the input data.

The input map defines all of the track's secondary inputs. Each secondary input is defined using a separate track input atom.

Each track input atom contains the following data elements.

Field descriptions

Size	A 32-bit integer that specifies the number of bytes in this track input atom.
Type	A 32-bit integer that identifies the atom type; this field must be set to `' in'` (note that the two leading bytes must be set to 0x00).
Atom ID	A 32-bit integer relating this track input atom to its secondary input. The value of this field corresponds to the index of the secondary input in the track reference atom. That is, the first secondary input corresponds to the track input atom with an Atom ID value of 1; the second to the track input atom with an Atom ID of 2, and so on.
Reserved	A 16-bit integer that must be set to 0.
Child count	A 16-bit integer specifying the number of child atoms in this atom.
Reserved	A 32-bit integer that must be set to 0.

The track input atom, in turn, may contain two other types of atoms: input type atoms and object ID atoms. The input type atom is required; it specifies how the data is to be interpreted.

The input type atom contains the following data elements.

Field descriptions

Size	A 32-bit integer that specifies the number of bytes in this input type atom.
Type	A 32-bit integer that identifies the atom type; this field must be set to `' ty'` (note that the two leading bytes must be set to 0x00).
Input type	A 32-bit integer that specifies the type of data that is to be received from the secondary data source. Table 1-3 lists valid values for this field.

Table 1-3 Input types

Input identifier	Description
1	A 3x3 transformation matrix to transform the track's location, scaling, and so on.
2	A QuickDraw clipping region to change the track's shape.
3	An 8.8 fixed point value indicating the relative sound volume. This is used for fading the volume.
4	A 16-bit integer indicating the sound balance level. This is used for panning the sound location.
5	A graphics mode record (32-bit integer indicating graphics mode, followed by an RGB color) to modify the track's graphics mode for visual fades.

Table 1-3 Input types (continued)

Input identifier	Description
6	A 3x3 transformation matrix to transform an object within the track's location, scaling, and so on.
7	A graphics mode record (32-bit integer indicating graphics mode, followed by an RGB color) to modify an object within the track's graphics mode for visual fades.
`'vide'`	Compressed image data for an object within the track.

If the input is operating on an object within the track (for example, a sprite within a sprite track), an object ID atom must be included in the track input atom to identify the object.

The object ID atom contains the following data elements.

Field descriptions

Size	A 32-bit integer that specifies the number of bytes in this object ID atom.
Type	A 32-bit integer that identifies the atom type; this field must be set to `'obid'`.
Object ID	A 32-bit integer identifying the object.

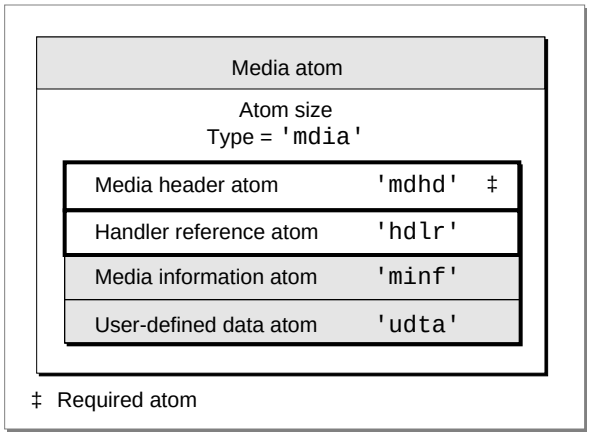
Media Atoms

Media atoms define a track's movie data. The media atom contains information that specifies the media handler component that is to interpret the media data, and it also specifies the data references.

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 data (`'udta'`). The only required atom in a media atom is the media header atom.

Figure 1-18 shows the layout of a media atom.

Figure 1-18 The layout of a media atom



Note
The handler reference atom tells you the 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," beginning on page 34, discusses how data may be stored in a media, using the video media format defined by Apple as an example. ◆

Media atoms contain the following data elements.

Field descriptions

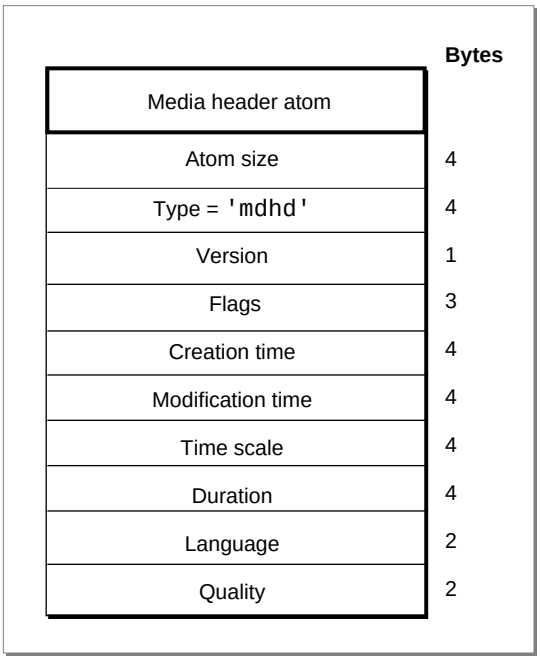
Size	A 32-bit integer that specifies the number of bytes in this media atom.
Type	A 32-bit integer that identifies the atom type; this field must be set to `'mdia'`.
Media header	The media header atom. This atom contains the standard media information. See the next section for details.
Media handler	The media handler atom. This atom identifies the media handler component that is to be used to interpret the media data. See "Handler Reference Atoms," beginning on page 33, for more information.
Media information	The media information atom. This atom contains media-type specific data for use by the media handler component. See "Media Information Atoms" beginning on page 34 for more information.
User data	The user data atom associated with this media. See "User Data Atoms" on page 57 for more information.

Media Header Atoms

The media header atom specifies the characteristics of a media, including time scale and duration. The media atom has an atom type of `'mdhd'`.

Figure 1-19 shows the layout of a media header atom.

Figure 1-19 The layout of a media header atom



Media header atoms contain the following data elements.

Field descriptions

Size	A 32-bit integer that specifies the number of bytes in this media header atom.
Type	A 32-bit integer that identifies the atom type; this field must be set to `'mdhd'`.
Version	One byte that specifies the version of this movie.
Flags	Three bytes of space for media header flags. Set this field to 0.
Creation time	A 32-bit integer that specifies (in seconds since midnight, January 1, 1904) when the media atom was created.
Modification time	A 32-bit 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 units of its time scale.
Language	A 16-bit integer that specifies the language code for this media. See “Basic Data Types” on page 75 for valid language codes.

Quality	A 16-bit integer that specifies the media's playback quality—that is, its suitability for playback in a given environment. See *Inside Macintosh: QuickTime* for details on playback quality.

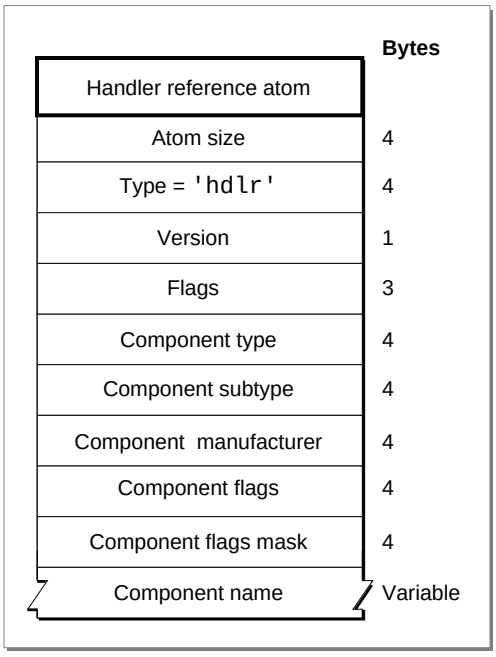
Handler Reference Atoms

The handler reference atom specifies the media handler component that is to be used to interpret the media's data. The handler reference atom has an atom type value of `'hdlr'`.

Historically, the handler reference atom was also used for data references. However, this use may now be ignored.

Figure 1-20 shows the layout of a handler reference atom.

Figure 1-20 The layout of a handler reference atom



Handler reference atoms contain the following data elements.

Field descriptions

Size	A 32-bit integer that specifies the number of bytes in this handler reference atom.
Type	A 32-bit integer that identifies the atom type; this field must be set to `'hdlr'`.
Version	A 1-byte specification of the version of this handler information.
Flags	A 3-byte space for handler information flags. Set this field to 0.

Component type A four-character code that identifies the type of the handler. Only two values are valid for this field: `'mhlr'` for media handlers and `'dhlr'` for data references.

Component subtype
A four-character code that identifies the type of the media handler or data handler. For media handlers, this field defines the type of data, for example, `'vide'` for video data or `'soun'` for sound data.

For data handlers, this field defines the data reference type. For example, a Component subtype value of `'alis'` identifies a file alias.

Component manufacturer
Reserved. Set to 0.

Component flags Reserved. Set to 0.

Component flags mask
Reserved. Set to 0.

Component name A Pascal string that specifies the name of the component—that is, the media handler used when this movie was created. This field may contain a zero-length (empty) string.

Media Information Atoms

Media information atoms (defined by the `'minf'` atom type) store handler-specific information for a track's media data. The media handler uses this information to map from media time to media data and to process the media data.

These atoms contain information that is specific to the type of data defined by the media. Further, 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.

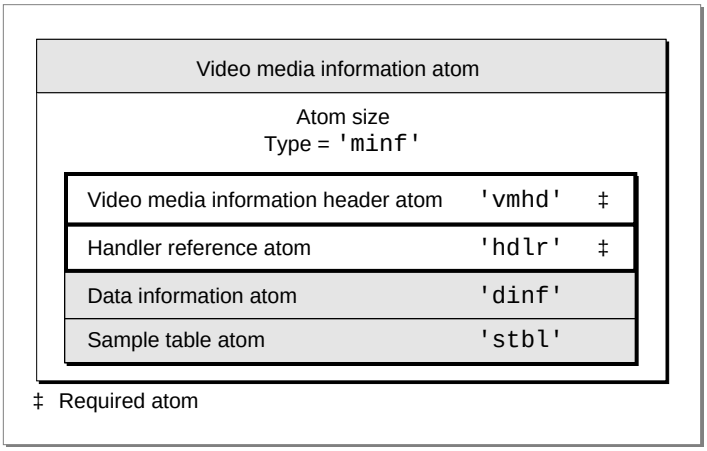
This section describes the atoms that store media information for the video (defined by the `'vmhd'` atom type), sound (defined by the `'smhd'` atom type), and base (defined by the `'gmhd'` atom type) portions of QuickTime movies.

Note
"Using Sample Atoms," beginning on page 56, 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. These atoms contain a number of other atoms that define specific characteristics of the video media data. Figure 1-21 shows the layout of a video media information atom.

Figure 1-21 The layout of a media information atom for video



The video media information atom contains the following data elements.

Field descriptions

Size — A 32-bit integer that specifies the number of bytes in this video media information atom.

Type — A 32-bit integer that identifies the atom type; this field must be set to `'minf'`.

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. "Handler Reference Atoms" beginning on page 33 discusses handler reference atoms. The handler uses the data information atom to understand the media's data references.

Data information — The data information atom, described in "Data Information Atoms" on page 41.

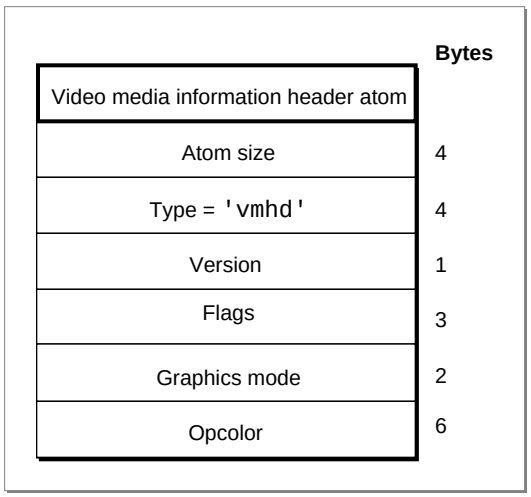
Sample table — The sample table atom, described in "Sample Table Atoms" on page 45.

Video Media Information Header Atoms

Video media information header atoms define specific color and graphics mode information.

Figure 1-22 shows the structure of a video media information header atom.

Figure 1-22 The layout of a media information header atom for video



The video media information header atom contains the following data elements.

Field descriptions

Size — A 32-bit integer that specifies the number of bytes in this video media information header atom.

Type — A 32-bit integer that identifies the atom type; this field must be set to `'vmhd'`.

Version — A 1-byte specification of the version of this video media information header atom.

Flags — A 3-byte space for video media information flags. There is one defined flag.

No lean ahead
: This is a compatibility flag that allows QuickTime to distinguish between movies created for QuickTime 1.0 and newer movies. You should always set this flag to 1, unless you are creating a movie intended for playback using version 1.0 of QuickTime. This flag's value is 0x0001.

Graphics mode — A 16-bit integer that specifies the transfer mode. The transfer mode specifies which Boolean operation QuickDraw should perform when drawing or transferring an image from one location to another.

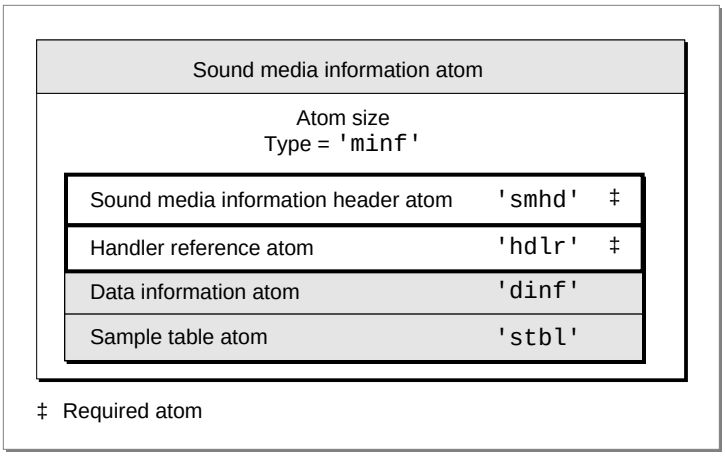
Opcolor — Three 16-bit values that specify the red, green, and blue colors for the transfer mode operation indicated in the Graphics mode field.

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

Sound Media Information Atoms

Sound media information atoms are the highest-level atoms in sound media. These atoms contain a number of other atoms that define specific characteristics of the sound media data. Figure 1-23 shows the layout of a sound media information atom.

Figure 1-23 The layout of a media information atom for sound



The sound media information atom contains the following data elements.

Field descriptions

Size — A 32-bit integer that specifies the number of bytes in this sound media information atom.

Type — A 32-bit integer that identifies the atom type; this field must be set to `'minf'`.

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. “Handler Reference Atoms” beginning on page 33 discusses handler reference atoms. The handler uses the data information atom to understand the media’s data references.

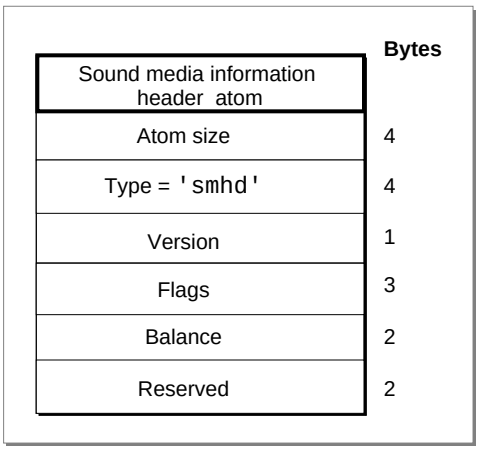
Data information — The data information atom, described in “Data Information Atoms” on page 41.

Sample table — The sample table atom, described in “Sample Table Atoms” on page 45.

Sound Media Information Header Atoms

The sound media information header atom (shown in Figure 1-24) stores the sound media’s control information, such as balance.

Figure 1-24 The layout of a sound media information header atom



The sound media information header atom contains the following data elements.

Field descriptions

Size — A 32-bit integer that specifies the number of bytes in this sound media information header atom.

Type — A 32-bit integer that identifies the atom type; this field must be set to `'smhd'`.

Version — A 1-byte specification of the version of this sound media information header atom.

Flags — A 3-byte space for sound media information flags. Set this field to 0.

Balance — A 16-bit 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 *Inside Macintosh: QuickTime* for more information about QuickTime sound balance values.

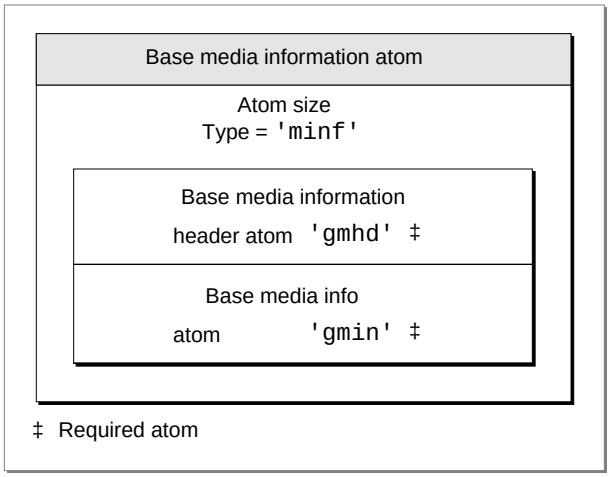
Reserved — Reserved for use by Apple. Set this field to 0.

Base Media Information Atoms

The base media information atom (shown in Figure 1-25) stores the media information for media types, such as text, MPEG, time code, and music.

Media types that are derived from the base media handler may add other atoms within the base media information atom, as appropriate. At present, the only media type that defines any additional atoms is the time code media. See “Time Code Media Information Atom” beginning on page 66 for more information about time code media.

Figure 1-25 The layout of the base media information atom



The base media information atom contains the following data elements.

Field descriptions

Size — A 32-bit integer that specifies the number of bytes in this base media information atom.

Type — A 32-bit integer that identifies the atom type; this field must be set to `'minf'`.

Base media information header — The base media information header atom (a required atom), which is described in the next section.

Base media info — The base media info atom contains control information that describes the media.

Base Media Information Header Atoms

The base media information header atom indicates that this media information atom pertains to a base media.

The base media information header atom contains the following data elements.

Field descriptions

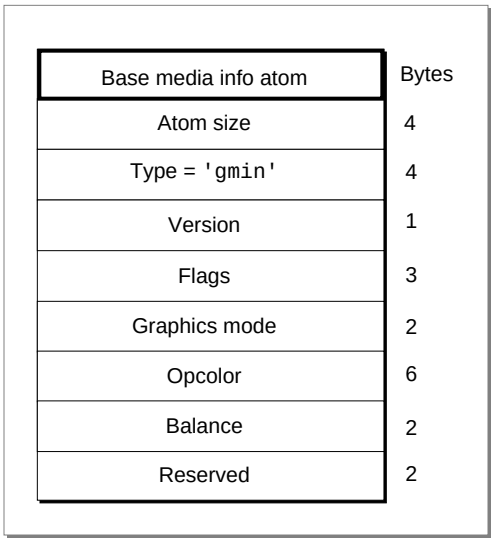
Size — A 32-bit integer that specifies the number of bytes in this base media information header atom.

Type — A 32-bit integer that identifies the atom type; this field must be set to `'gmhd'`.

Base Media Info Atoms

The base media info atom, contained in the base media information atom, defines the media's control information, including graphics mode and balance information. Figure 1-26 shows the layout of a base media info atom.

Figure 1-26 The layout of the base media info atom



The base media info atom contains the following data elements.

Field descriptions

Size	A 32-bit integer that specifies the number of bytes in this base media information header atom.
Type	A 32-bit integer that identifies the atom type; this field must be set to `'gmin'`.
Version	A 1-byte specification of the version of this base media information header atom.
Flags	A 3-byte space for base media information flags. Set this field to 0.
Graphics mode	A 16-bit integer that specifies the transfer mode. The transfer mode specifies 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.
Balance	A 16-bit integer that specifies the sound balance of this 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 *Inside Macintosh: QuickTime* for more information about QuickTime sound balance values.
Reserved	Reserved for use by Apple. Set this field to 0.

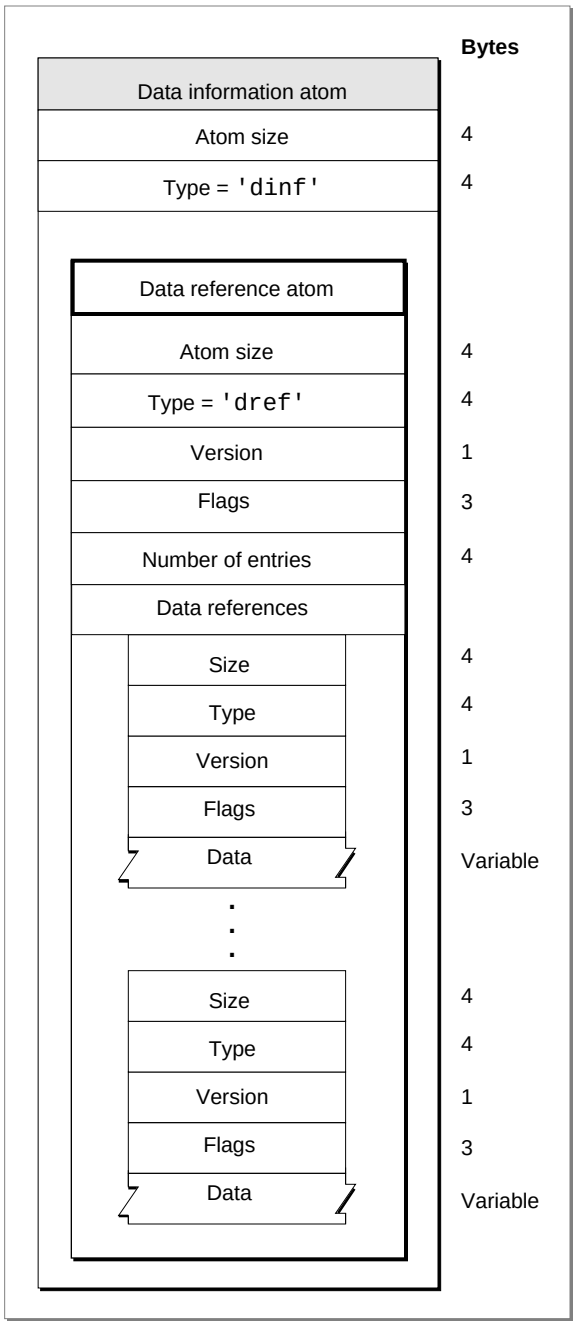
For comprehensive details on transfer modes and opcolors and their values in QuickDraw, see *Inside Macintosh: Imaging*.

Data Information Atoms

The handler reference atom (described in "Handler Reference Atoms," beginning on page 33) contains information specifying the data handler component that provides access to the media data. The data handler component uses the data information atom to interpret the media's data. Data information atoms have an atom type value of `'dinf'`.

Figure 1-27 shows the layout of a data information atom.

Figure 1-27 The layout of a data information atom



The data information atom contains the following data elements.

Field descriptions

Size A 32-bit integer that specifies the number of bytes in this data information atom.

Type	A 32-bit integer that identifies the atom type; this field must be set to `'dinf'`.
Data references	A data reference atom, described in the next section, contains the data references.

Data Reference Atoms

Data reference atoms contain tabular data that instructs the data handler component how to access the media's data. Figure 1-27 shows the data reference atom.

The data reference atom contains the following data elements.

Field descriptions

Size	A 32-bit integer that specifies the number of bytes in this data reference atom.
Type	A 32-bit integer that identifies the atom type; this field must be set to `'dref'`.
Version	A 1-byte specification of the version of this data reference atom.
Flags	A 3-byte space for data reference flags. Set this field to 0.
Number of entries	A 32-bit integer containing the count of data references that follow.
Data references	An array of data references.

Each data reference is formatted like an atom and contains the following data elements.

Field descriptions

Size	A 32-bit integer that specifies the number of bytes in these data references.
Type	A 32-bit integer that specifies the type of the data in the data references. Table 1-4 lists valid Type values.
Version	A 1-byte specification of the version of these data references.
Flags	A 3-byte space for data reference flags. There is one defined flag.
Self reference	This flag indicates that the media's data is in the same file as the movie atom. On the Macintosh, and other file systems with multifork files, set this flag to 1 even if the data resides in a different fork from the movie atom. This flag's value is 0x0001.
Data references	The data reference information.

Table 1-4 shows the currently defined data reference types that may be stored in a movie.

Table 1-4 Data reference types

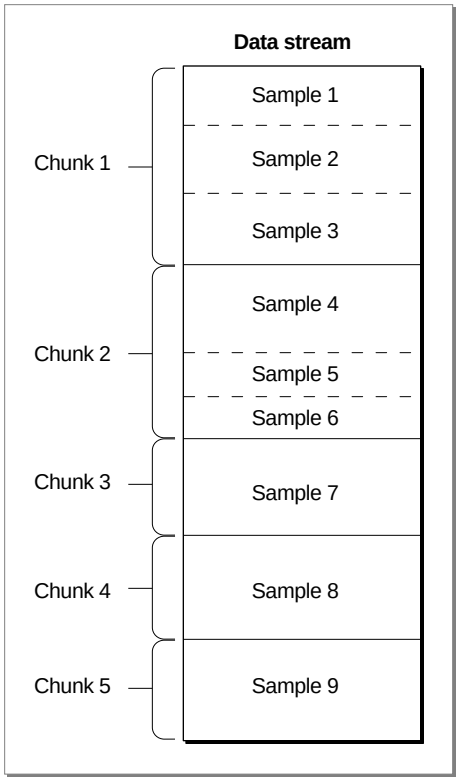
Data reference type	Description
`'alis'`	Data reference is a Macintosh alias. An alias contains information about the file, including its full path name. For more information, see *Inside Macintosh: Files*.
`'rsrc'`	Data reference is a Macintosh alias. Appended to the end of the alias is the resource type (stored as a 32-bit unsigned integer) and ID (stored as a 16-bit signed integer) to use within the specified file.

Sample Atoms

QuickTime stores media data in samples. A sample is a single element in a sequence of time-ordered data. Samples are stored in the media, and they may have varying durations.

Figure 1-28 shows the way that samples are stored in a series of **chunks** in a media.

Figure 1-28 Samples 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 individual samples within a chunk may have different sizes from one another.

One way to describe a sample is to use a sample table atom. The sample table atom acts as a storehouse of information about the samples and contains a number of different types of atoms. The various atoms contain information that allows the media handler to parse the samples in the proper order. This approach enforces an ordering of the samples without requiring that the sample data be stored sequentially with respect to movie time in the actual data stream.

The next section discusses the sample table atom. Subsequent sections discuss each of the atoms that may reside in a sample table atom.

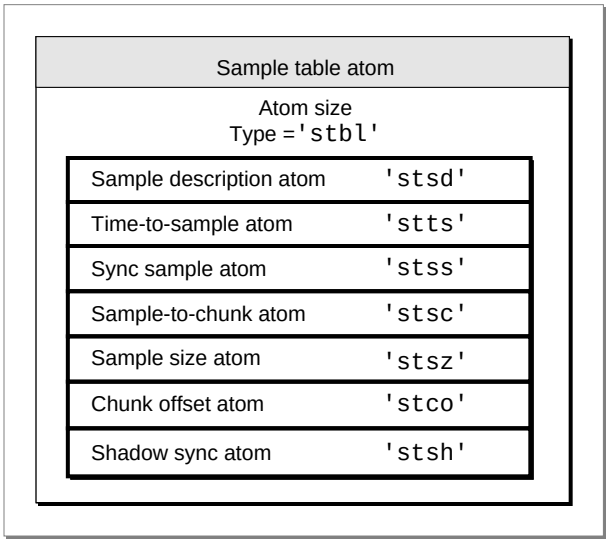
Sample Table Atoms

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 data 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 1-29 shows the layout of a sample table atom.

Figure 1-29 The layout of a sample table atom



The sample table atom contains the following data elements.

Field descriptions

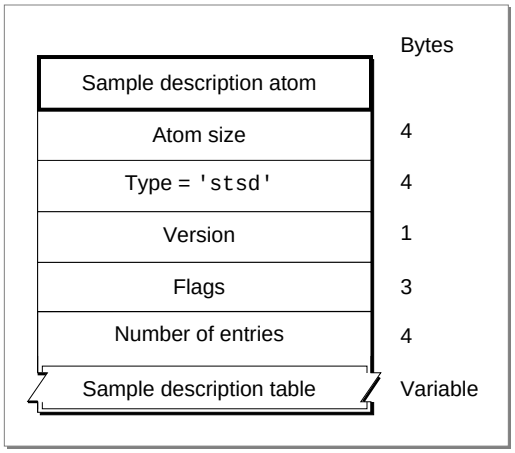
Size	A 32-bit integer that specifies the number of bytes in this sample table atom.
Type	A 32-bit integer that identifies the atom type; this field must be set to `'stbl'`.
Sample description	The sample description atom, described in the next section.
Time-to-sample	The time-to-sample atom, described in "Time-to-Sample Atoms," beginning on page 48.
Sync sample	The sync sample atom, described in "Sync Sample Atoms," beginning on page 50.
Sample-to-chunk	The sample-to-chunk atom, described in "Sample-to-Chunk Atoms," beginning on page 51.
Sample size	The sample size atom, described in "Sample Size Atoms," beginning on page 53.
Chunk offset	A chunk offset atom, described in "Chunk Offset Atoms," beginning on page 55.
Shadow sync	The shadow sync atom. This atom is obsolete.

Sample Description Atoms

The sample description atom stores information that allows you to decode samples in the media. The data stored in the sample description varies, depending on the media type. For example, in the case of video media, the sample descriptions are image description structures. The sample description information for each media type is explained later in this document, in “Media Data Atom Types” beginning on page 59.

Figure 1-30 shows the layout of a sample description atom.

Figure 1-30 The layout of a sample description atom



The sample description atom has an atom type of `'stsd'`. The sample description atom contains a table of sample descriptions. A media may have one or more sample descriptions, depending upon the number of different encoding schemes used in the media and on the number of files used to store the data. 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,” beginning on page 51).

The sample description atom contains the following data elements.

Field descriptions

Size	A 32-bit integer that specifies the number of bytes in this sample description atom.
Type	A 32-bit integer that identifies the atom type; this field must be set to `'stsd'`.
Version	A 1-byte specification of the version of this sample description atom.
Flags	A 3-byte space for sample description flags. Set this field to 0.
Number of entries	A 32-bit integer containing the number of sample descriptions that follow.

Sample description table
: An array of sample descriptions.

While the exact format of the sample description varies by media type, the first four fields of every sample description table are the same.

Field descriptions

Sample description size
: A 32-bit integer indicating the number of bytes in the sample description.

Data format
: A 32-bit integer indicating the format of the stored data. This depends on the media type, but is usually either the compression format or the media type.

Reserved
: Six bytes that must be set to 0.

Data reference index
: A 16-bit integer that contains the index of the data reference to use to retrieve data associated with samples that use this sample description. Data references are stored in data reference atoms.

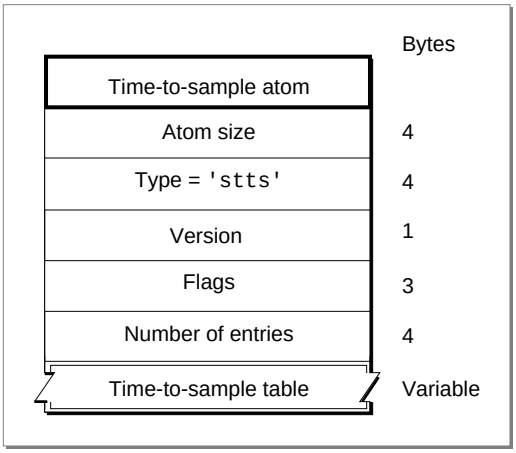
Time-to-Sample Atoms

Time-to-sample atoms store duration information for a media's samples, providing a mapping from a time in the 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 time in a media by examining the time-to-sample atom table, which is contained in a time-to-sample atom.

Figure 1-31 shows the layout of a time-to-sample atom.

Figure 1-31 The layout of a time-to-sample atom

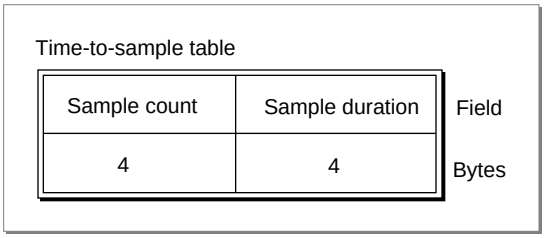


The time-to-sample atom contains the following data elements.

Field descriptions

Size	A 32-bit integer that specifies the number of bytes in this time-to-sample atom.
Type	A 32-bit integer that identifies the atom type; this field must be set to `'stts'`.
Version	A 1-byte specification of the version of this time-to-sample atom.
Flags	A 3-byte space for time-to-sample flags. Set this field to 0.
Number of entries	A 32-bit integer containing the number of entries in the time-to-sample table.
Time-to-sample table	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 a time-to-sample table is shown in Figure 1-32.

Figure 1-32 The layout of a time-to-sample table



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

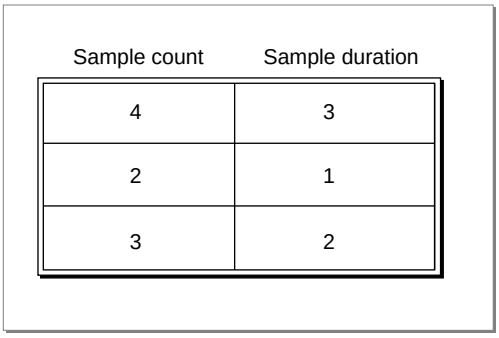
Field descriptions

Sample count	A 32-bit integer that specifies the number of consecutive samples that have the same duration.
Sample duration	A 32-bit integer that specifies the duration of each sample.

Entries in the table describe 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 and the count would be equal to the number of samples.

Figure 1-33 presents an example of a time-to-sample table that is based on the chunked media data shown in Figure 1-28 on page 45. That data stream contains a total of nine samples that correspond in count and duration to the entries in the table shown here. 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 1-33 An example of a time-to-sample table



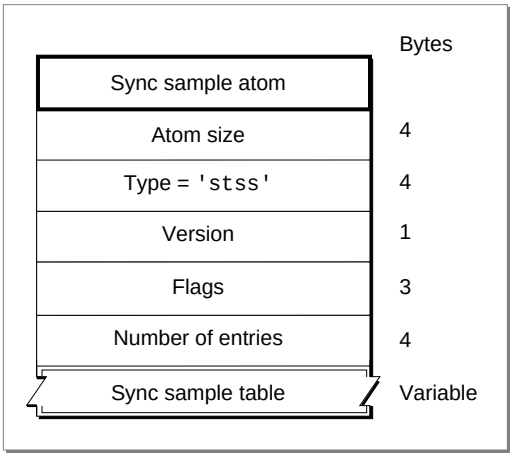
Sync Sample Atoms

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. 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 1-34 shows the layout of a sync sample atom.

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

Figure 1-34 The layout of a sync sample atom

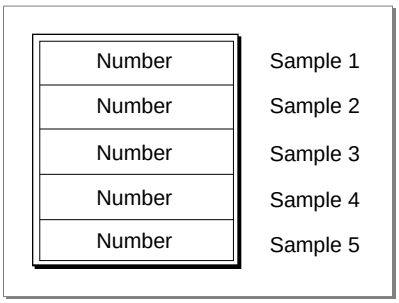


The sync sample atom contains the following data elements.

Field descriptions

Size	A 32-bit integer that specifies the number of bytes in this sync sample atom.
Type	A 32-bit integer that identifies the atom type; this field must be set to `'stss'`.
Version	A 1-byte specification of the version of this sync sample atom.
Flags	A 3-byte space for sync sample flags. Set this field to 0.
Number of entries	A 32-bit integer containing the number of entries in the sync sample table.
Sync sample table	A table of sample numbers; each sample number corresponds to a key frame. Figure 1-35 shows the layout of a sync sample table.

Figure 1-35 The layout of a sync sample table

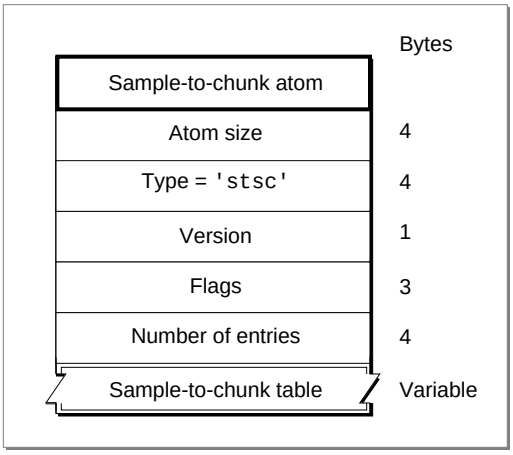


Sample-to-Chunk Atoms

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.

Sample-to-chunk atoms have an atom type of `'stsc'`. The sample-to-chunk atom contains a table that maps samples to chunks in the media data stream. Figure 1-36 shows the layout of a sample-to-chunk atom. By examining the sample-to-chunk atom, you can determine the chunk that contains a specific sample.

Figure 1-36 The layout of a sample-to-chunk atom



The sample-to-chunk atom contains the following data elements.

Field descriptions

Size — A 32-bit integer that specifies the number of bytes in this sample-to-chunk atom.

Type — A 32-bit integer that identifies the atom type; this field must be set to `'stsc'`.

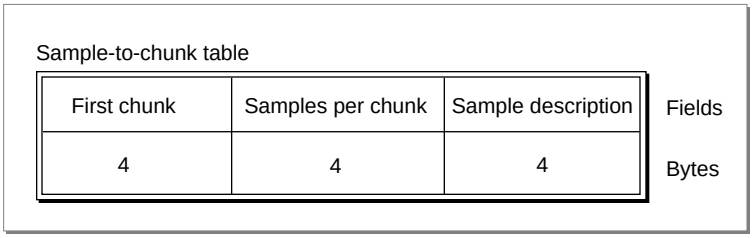
Version — A 1-byte specification of the version of this sample-to-chunk atom.

Flags — A 3-byte space for sample-to-chunk flags. Set this field to 0.

Number of entries — A 32-bit integer containing the number of entries in the sample-to-chunk table.

Sample-to-chunk table — A table that maps samples to chunks. Figure 1-37 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 1-37 The layout of a sample-to-chunk table



You define a sample-to-chunk table by specifying the following data elements.

Field descriptions

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 for the sample. For details on sample description atoms, see "Sample Description Atoms," beginning on page 47.

Figure 1-38 shows an example of a sample-to-chunk table that is based on the data stream shown in Figure 1-28 on page 45.

Figure 1-38 An example of a sample-to-chunk table

Sample-to-chunk table

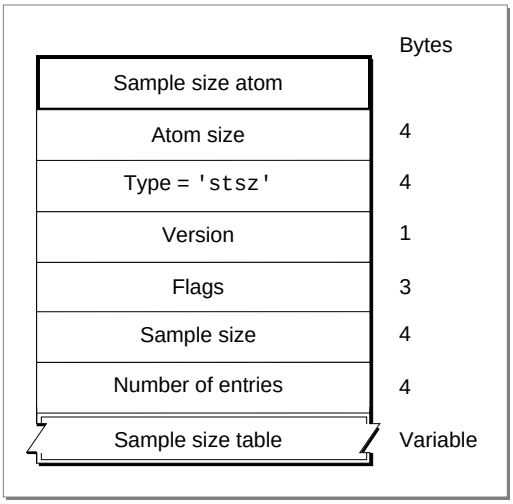
First chunk	Samples per chunk	Sample description ID
1	3	23
3	1	23
5	1	24

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. 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 specify the size of each sample in the media. Sample size atoms have an atom type of `'stsz'`. Figure 1-39 shows the layout of a sample size atom.

Figure 1-39 The layout of a sample size atom



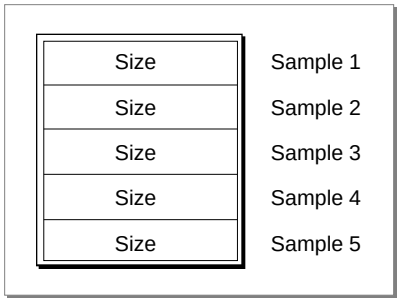
The sample size atom contains the following data elements.

Field descriptions

Size	A 32-bit integer that specifies the number of bytes in this sample size atom.
Type	A 32-bit integer that identifies the atom type; this field must be set to `'stsz'`.
Version	A 1-byte specification of the version of this sample size atom.
Flags	A 3-byte space for sample size flags. Set this field to 0.
Sample size	A 32-bit integer specifying the sample size. If all the samples are the same size, this field contains that size value. If this field is set to 0, then the samples have different sizes, and those sizes are stored in the sample size table.
Number of entries	A 32-bit integer containing the number of entries in the sample size table.
Sample size table	A table containing the sample size information. The sample size table contains an entry for every sample in the media's data stream. Each table entry contains a size field. The size field contains the size, in bytes, of the sample in question. 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.

Figure 1-40 shows a sample size table.

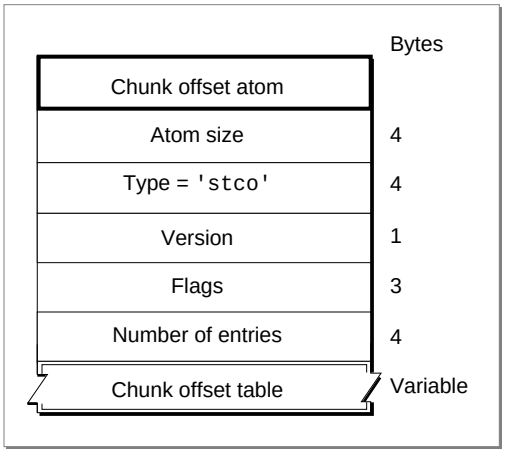
Figure 1-40 An example of a sample size table



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 1-41) contains a table of offset information.

Figure 1-41 The layout of a chunk offset atom



The chunk offset atom contains the following data elements.

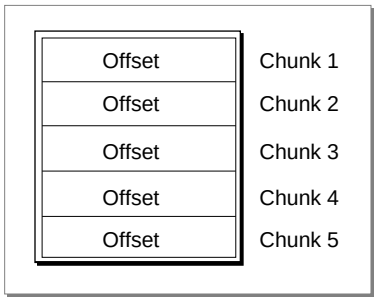
Field descriptions

Size	A 32-bit integer that specifies the number of bytes in this chunk offset atom.
Type	A 32-bit integer that identifies the atom type; this field must be set to `'stco'`.
Version	A 1-byte specification of the version of this chunk offset atom.
Flags	A 3-byte space for chunk offset flags. Set this field to 0.
Number of entries	A 32-bit integer containing the number of entries in the chunk offset table.

Chunk offset table — A chunk offset table consisting of an array of offset values. There is one table entry for each chunk in the media. The offset contains the byte offset from the beginning of the data stream to the chunk. 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.

Figure 1-42 shows an example of a chunk offset table

Figure 1-42 An example of a chunk offset table



Using Sample 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 section, "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 section, "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 QuickTime displays a movie or track, it 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 and the sample's size 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 and the sample's size by using the sample size atom.

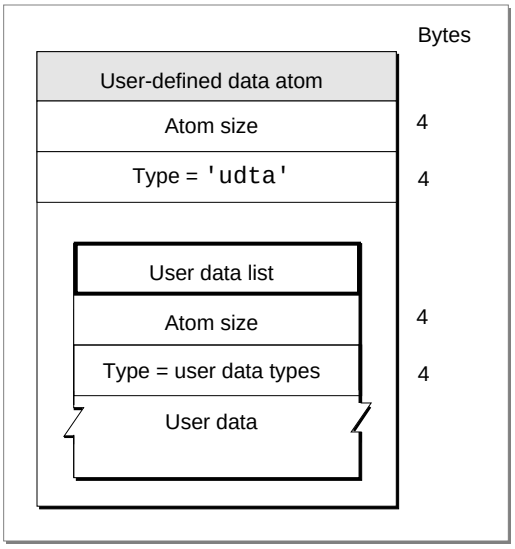
User Data Atoms

User data atoms allow you to define and store arbitrary data associated with a QuickTime object, such as a movie, track, or media. The user data atom has an atom type of `'udta'`.

Inside the user data atom is a list of atoms describing each piece of user data. User data provides a simple way to extend what is stored in a QuickTime movie. For example, you may use user data atoms to store a movie's window position, playback characteristics, or creation information.

Figure 1-43 shows the layout of a user data atom.

Figure 1-43 The layout of a user data atom



The user data atom contains the following data elements.

Field descriptions

Size	A 32-bit integer that specifies the number of bytes in this user data atom.
Type	A 32-bit integer that identifies the atom type; this field must be set to `'udta'`.
User data list	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, 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. Table 1-5 lists the currently defined list entry types.

Table 1-5 User data list entry types

List entry type	Description
`'©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
`'WLOC'`	Default window location for movie. Two 16 bit values, {x,y}.
`'name'`	Name of object.
`'LOOP'`	Long integer indicating looping style. 0 for none, 1 for looping, 2 for palindrome looping.
`'SelO'`	Play selection only. Byte indicating that only the selected area of the movie be played.
`'AllF'`	Play all frames. Byte indicating that all frames of video should be played, regardless of timing.

All user data list entries whose type begins with the '©' character (ASCII 169), are defined to be international text. These list entries must contain a list of text strings with associated language codes. By storing multiple versions of the same text, a single user data text item can contain translations for different languages.

The list of text strings uses a small integer atom format, which is identical to the QuickTime atom format, except that it uses 16-bit values for size and type instead of 32-bit values. The first value is the size of the string, including the size and type, and the second value is the language code for the string.

Media Data Atom Types

QuickTime uses atoms of different types to store different types of media data—video media for video data, sound media for audio data, and so on. The following sections discuss each of these different media data atom types.

Video Media

Video media is used to store compressed and uncompressed image data in QuickTime movies. It has a media type of `'vide'`.

Video Sample Description

The video sample description contains information that defines how to interpret video media data. This sample description is based on the standard sample description header, as described in “Sample Table Atoms” beginning on page 45.

The Data format field indicates the type of compression that was used to compress the image data. Table 1-6 shows some of the formats currently supported.

Table 1-6 Image compression formats

Compression type	Description
`'cvid'`	Cinepak
`'jpeg'`	JPEG
`'raw '`	Uncompressed RGB
`'YUV2'`	Uncompressed YUV422
`'smc '`	Graphics
`'rle '`	Animation
`'rpza'`	Apple Video
`'kpcd'`	Kodak Photo CD
`'qdgx'`	QuickDraw GX

Table 1-6 Image compression formats (continued)

Compression type	Description
`'mpeg'`	MPEG still image
`'mjpa'`	Motion-JPEG (Format A)
`'mjpb'`	Motion-JPEG (Format B)

The video media handler also adds some of its own fields to the sample description. For more information about each of these fields, see the discussion of image descriptions in *Inside Macintosh: QuickTime*.

Field descriptions

Version	A 16-bit integer indicating the version number of the compressed data. Usually this is set to 0, unless a compressor has changed its data format.
Revision level	A 16-bit integer that must be set to 0.
Vendor	A 32-bit integer that specifies the developer of the compressor that generated the compressed data. Often this field contains `'appl'` to indicate Apple Computer, Inc.
Temporal quality	A 32-bit integer containing a value from 0 to 1023 indicating the degree of temporal compression.
Spatial quality	A 32-bit integer containing a value from 0 to 1023 indicating the degree of spatial compression.
Width	A 16-bit integer that specifies the width of the source image in pixels.
Height	A 16-bit integer that specifies the height of the source image in pixels.
Horizontal resolution	A 32-bit fixed point number containing the horizontal resolution of the image in pixels per inch.
Vertical resolution	A 32-bit fixed point number containing the vertical resolution of the image in pixels per inch.
Data size	A 32-bit integer that must be set to 0.
Frame count	A 16-bit integer that indicates how many frames of compressed data are stored in each sample. Usually set to 1.
Compressor name	A 32-byte Pascal string containing the name of compressor that created the image, such as `"jpeg"`.
Depth	A 16-bit integer that indicates the pixel depth of the compressed image. Values of 1, 2, 4, 8,16, 24, and 32 indicate the depth of color images. The value of 32 should be used only if the image contains an alpha channel. Values of 34, 36, and 40 indicate 2-, 4-, and 8-bit grayscale, respectively, for grayscale images.
Color table ID	A 16-bit integer that identifies which color table to use. If this field is set to –1, the default color table should be used for the specified depth. For all depths below 16 bits per pixel, this indicates a

standard Macintosh color table for the specified depth. Depths of 16, 24, and 32 have no color table.

If the color table ID is set to 0, a color table is contained within the sample description itself. The color table immediately follows the Color table ID field in the sample description. See "Color Table Atoms" on page 14 for a complete description of a color table.

Video sample descriptions can be extended by appending other atoms. These atoms are placed after the color table, if one is present. These extensions to the sample description may contain display hints for the decompressor or may simply carry additional information associated with the images. Table 1-7 lists the currently defined extensions to video sample descriptions.

Table 1-7 Video sample description extensions

Extension type	Description
`'gama'`	A 32-bit fixed point number indicating the gamma level at which the image was captured. The decompressor can use this value to gamma-correct at display time.
`'fiel'`	A 2-byte value indicating the number of video fields in the data stream, and how those fields are to be used. The first byte specifies the field count, and may be set to 1 or 2. The second byte defines the field dominance, as follows: 0–Field dominance unknown 1–Top field is first, temporally 2–Bottom field is first, temporally This information is used by applications that may be modifying decompressed image data, or by decompressor components to determine field display order.
`'mjqt'`	The default quantization table for a Motion JPEG data stream.
`'mjht'`	The default Huffman table for a Motion JPEG data stream.

Video Sample Data

The format of the data stored in video samples is completely dependent on the type of the compressed data stored in the video sample description. The following sections discuss each of the video encoding schemes supported by QuickTime.

Uncompressed RGB

Uncompressed RGB data is stored in a variety of different formats. The format used depends on the Depth field of the video sample description. For all depths, the image data is padded on each scan line to ensure that each scan line begins on an even byte boundary.

- For depths of 1, 2, 4, and 8, the values stored are indexes into the color table specified in the Color table id field.
- For a depth of 16, the pixels are stored as 5-5-5 RGB values with the high bit of each 16-bit integer set to 0.
- For a depth of 24, the pixels are stored packed together in RGB order.
- For a depth of 32, the pixels are stored with an 8-bit alpha channel, followed by 8-bit RGB components.

Uncompressed YUV

Uncompressed YUV data is stored as YUV422 data. Two pixels are combined into a single 32-bit integer stored in YUYV order.

JPEG

QuickTime stores JPEG images according to the rules described in the ISO JPEG specification, document number DIS 10918-1.

Motion JPEG

Motion JPEG (M-JPEG) is a variant of the ISO JPEG specification for use with digital video streams. Instead of compressing an entire image into a single bitstream, M-JPEG compresses each video field separately, returning the resulting JPEG bitstreams consecutively in a single frame.

There are two flavors of M-JPEG currently in use. These two formats differ based on their use of markers. **M-JPEG format A** supports markers; **M-JPEG format B** does not. The following paragraphs describe how QuickTime stores M-JPEG sample data.

Each field of M-JPEG Format A fully complies with the ISO JPEG specification, and therefore supports application markers. QuickTime uses the app 1 marker to store control information, as follows (all of the fields are 32-bit integers):

Field descriptions

Reserved	Contents unpredictable; should be set to 0.
Tag	Identifies the data type; this field must be set to `'mjpg'`.
Field size	Contains the actual size of the image data for this field, in bytes.
Padded field size	Contains the size of the image data, including pad bytes. Some video hardware may append pad bytes to the image data; this field, along with the Field size field, allows you to compute how many pad bytes were added.
Offset to next field	Specifies the offset, in bytes, from the start of the field data to the start of the next field in the bitstream. This field should be set to 0 in the second field's marker data.
Quantization table offset	Specifies the offset, in bytes, from the start of the field data to the quantization table marker. If this field is set to 0, check the image description for a default quantization table (see "Video Sample Description" beginning on page 59 for details).

Huffman table offset
: Specifies the offset, in bytes, from the start of the field data to the Huffman table marker. If this field is set to 0, check the image description for a default Huffman table (see "Video Sample Description" beginning on page 59 for details).

Start of image offset
: Specifies the offset from the start of the field data to the start of image marker. This field should never be set to 0.

M-JPEG Format B does not support markers. In place of the marker, therefore, QuickTime inserts a header at the beginning of the bitstream. Again, all of the fields are 32-bit integers.

Field descriptions

Reserved	Contents unpredictable; should be set to 0.
Tag	Identifies the data type; this field must be set to `'mjpg'`.
Field size	Contains the actual size of the image data for this field, in bytes.
Padded field size	Contains the size of the image data, including pad bytes. Some video hardware may append pad bytes to the image data; this field, along with the Field size field, allows you to compute how many pad bytes were added.
Offset to next field	Specifies the offset, in bytes, from the start of the field data to the start of the next field in the bitstream. This field should be set to 0 in the second field's header data.
Quantization table offset	Specifies the offset, in bytes, from the start of the field data to the quantization table. If this field is set to 0, check the image description for a default quantization table (see "Video Sample Description" beginning on page 59 for details).
Huffman table offset	Specifies the offset, in bytes, from the start of the field data to the Huffman table. If this field is set to 0, check the image description for a default Huffman table (see "Video Sample Description" beginning on page 59 for details).
Start of image offset	Specifies the offset from the start of the field data to the field's image data. This field should never be set to 0.
Reserved	Must be set to 0.
Reserved	Must be set to 0.

The M-JPEG Format B header must be a multiple of 16 in size. When you add pad bytes to the header, set them to 0.

Note

Because this format does not support markers, there is no need to stuff the bitstream with null bytes (0x00) after data bytes that are set to 0xFF. ◆

MPEG Still Images

QuickTime stores MPEG still images according to the Blue Book specification for Enhanced CDs.

Sound Media

Sound media is used to store compressed and uncompressed audio data in QuickTime movies. It has a media type of `'soun'`.

Sound Sample Description

The sound sample description contains information that defines how to interpret sound media data. This sample description is based on the standard sample description header, as described in "Sample Table Atoms" beginning on page 45.

The data format field contains the format of the audio data. Table 1-8 shows some of the current formats.

Table 1-8 Sound data format types

Data format	Description
`'raw '`	Samples are stored uncompressed in offset-binary format (values range from 0 to 255; 128 is silence).
`'twos'`	Samples are stored uncompressed, in twos-complement format (sample values range from –128 to 127 for 8-bit audio, and -32768 to 32767 for 1-bit audio; 0 is always silence).
`'MAC3'`	Samples have been compressed using MACE 3:1.
`'MAC6'`	Samples have been compressed using MACE 6:1.
`'ima4'`	Samples have been compressed using IMA 4:1.
`'µlaw'`	Samples have been compressed using µLaw 2:1.

The sound media handler also adds some of its own fields to the sample description. For more information about each of these fields, see *Inside Macintosh: QuickTime*.

Field descriptions

Version — A 16-bit integer that must be set to 0.

Revision level — A 16-bit integer that must be set to 0.

Vendor — A 32-bit integer that must be set to 0.

Number of channels — A 16-bit integer that indicates the number of sound channels used by the sound sample. Set this field to 1 for monaural sounds; set it to 2 for stereo sounds.

Sample size — A 16-bit integer that specifies the number of bits in each uncompressed sound sample. Set this field to 8 for 8-bit sound and

	to 16 for 16-bit sound. Sound stored in `'twos'` format may contain 32-bit samples.
Compression ID	A 16-bit integer that must be set to 0.
Packet size	A 16-bit integer that must be set to 0.
Sample rate	A 32-bit unsigned fixed-point number that indicates the rate at which the sound samples were obtained. This number should match the media's time scale.

Sound Sample Data

The format of data stored in sound samples is completely dependent on the type of the compressed data stored in the sound sample description. The following sections discuss each of the formats supported by QuickTime.

Uncompressed 8-bit Sound

Eight-bit audio may be stored in either twos-complement or offset-binary encodings. In either case, if the data is in stereo, the left and right channels are interleaved.

Uncompressed 16-bit Sound

Sixteen-bit audio may be stored in either twos-complement or offset-binary encodings. However, twos-complement encoding is preferred. In either case, if the data is in stereo, the left and right channels are interleaved.

MACE 3:1 and 6:1

Not currently documented by Apple Computer. These are 8-bit formats.

IMA 4:1

The IMA encoding scheme is based on a standard developed by the International Multimedia Association for pulse code modulation (PCM) audio compression. QuickTime uses a slight variation of the format to allow for random access. IMA is a 16-bit audio format.

µLaw 2:1

The µLaw encoding scheme is used on North American and Japanese phone systems, and is coming into use for voice data interchange, and in PBXs, voice-mail systems, and Internet Talk Radio (via MIME). In µLaw encoding, 14 bits of linear sample data are reduced to 8 bits of logarithmic data.

Time Code Media

Time code media is used to store time code data in QuickTime movies. It has a media type of `'tmcd'`.

Time Code Sample Description

The time code sample description contains information that defines how to interpret time code media data. This sample description is based on the standard sample description header, as described in "Sample Table Atoms" beginning on page 45.

The Data format field in the sample description is always set to `'tmcd'`.

The time code media handler also adds some of its own fields to the sample description.

Field descriptions

Reserved — A 32-bit integer that is reserved for future use. Set this field to 0.

Flags — A 32-bit integer containing flags that identify some time code characteristics. The following flags are defined.

- Drop frame — Indicates whether the time code is drop frame. Set it to 1 if the time code is drop frame. This flag's value is 0x0001.
- 24 hour max — Indicates whether the time code wraps after 24 hours. Set it to 1 if the time code wraps. This flag's value is 0x0002.
- Negative times OK — Indicates whether negative time values are allowed. Set it to 1 if the time code supports negative values. This flag's value is 0x0004.
- Counter — Indicates whether the time value corresponds to a tape counter value. Set it to 1 if the time code values are tape counter values. This flag's value is 0x0008.

Time scale — A 32-bit integer that specifies the time scale for interpreting the Frame duration field.

Frame duration — A 32-bit integer that indicates how long each frame lasts in real time.

Number of frames — An 8-bit integer that contains the number of frames per second for the time code format. If the time is a counter, this is the number of frames for each counter tick.

Reserved — A 24-bit quantity that must be set to 0.

Source reference — A user data atom containing information about the source video tape. The only currently used user data list entry is the `'name'` field. This entry contains an international text item specifying the name of the source tape.

Time Code Media Information Atom

The time code media also requires a media information atom. This atom contains information governing how the time code text is displayed. This media information atom is stored in a base media information atom (see "Base Media Information Atoms" beginning on page 38 for more information). The type of the time code media information atom is `'tcmi'`.

The time code media information atom contains the following data elements.

Field descriptions

Size	A 32-bit integer that specifies the number of bytes in this time code media information atom.
Type	A 32-bit integer that identifies the atom type; this field must be set to `'tcmi'`.
Version	A 1-byte specification of the version of this time code media information atom.
Flags	A 3-byte space for time code media information flags. Set this field to 0.
Text font	A 16-bit integer that indicates the font to use. Set this field to 0 to use the system font. If the Font name field contains a valid name, ignore this field.
Text face	A 16-bit integer that indicates the font's style. Set this field to 0 for normal text. You can enable other style options by using one or more of the following bit masks:

0x0001	Bold
0x0002	Italic
0x0004	Underline
0x0008	Outline
0x0010	Shadow
0x0020	Condense
0x0040	Extend

Text size	A 16-bit integer that specifies the point size of the time code text.
Text color	A 48-bit RGB color value for the time code text.
Background color	A 48-bit RGB background color for the time code text.
Font name	A Pascal string specifying the name of the time code text's font.

Time Code Sample Data

There are two different sample data formats used by time code media.

If the Counter flag is set to 1 in the time code sample description, the sample data is a counter value. Each sample contains a 32-bit integer counter value.

If the Counter flag is set to 0 in the time code sample description, the sample data format is a time code record, as follows.

Field descriptions

Hours	An 8-bit unsigned integer that indicates the starting number of hours.
Negative	A 1-bit value indicating the time's sign. If the bit is set to 1, the time code record value is negative.
Minutes	A 7-bit integer that contains the starting number of minutes.
Seconds	An 8-bit unsigned integer indicating the starting number of seconds.

Frames An 8-bit unsigned integer that specifies the starting number of frames. This field's value cannot exceed the value of the Number of frames field in the time code sample description.

Text Media

Text media is used to store text data in QuickTime movies. It has a media type of `'text'`.

Text Sample Description

The time code sample description contains information that defines how to interpret time code media data. This sample description is based on the standard sample description header, as described in "Sample Table Atoms" beginning on page 45.

The Data format field in the sample description is always set to `'text'`.

The text media handler also adds some of its own fields to the sample description.

Field descriptions

Display flags A 32-bit integer containing flags that describe how the text should be drawn. The following flags are defined.

Don't auto scale
: Controls text scaling. If this flag is set to 1, the text media handler reflows the text instead of scaling when the track is scaled. This flag's value is 0x0002.

Use movie background color
: Controls background color. If this flag is set to 1, the text media handler ignores the Background color field in the text sample description and uses the movie's background color instead. This flag's value is 0x0008.

Scroll in
: Controls text scrolling. If this flag is set to 1, the text media handler scrolls the text in until the last of the text is in view. This flag's value is 0x0020.

Scroll out
: Controls text scrolling. If this flag is set to 1, the text media handler scrolls the text out until the last of the text is gone. This flag's value is 0x0040.

Horizontal scroll
: Controls text scrolling. If this flag is set to 1, the text media handler scrolls the text horizontally; otherwise, it scrolls the text vertically. This flag's value is 0x0080.

Reverse scroll
: Controls text scrolling. If this flag is set to 1, the text media handler scrolls down (if scrolling vertically) or backward (if scrolling horizontally; horizontal scrolling also depends upon text justification). This flag's value is 0x0100.

	Continuous scroll	Controls text scrolling. If this flag is set to 1, the text media handler displays new samples by scrolling out the old ones. This flag's value is 0x0200.
	Drop shadow	Controls drop shadow. If this flag is set to1, the text media handler displays the text with a drop shadow. This flag's value is 0x1000.
	Anti-alias	Controls anti-aliasing. If this flag is set to 1, the text media handler uses anti-aliasing when drawing text. This flag's value is 0x2000.
	Key text	Controls background color. If this flag is set to 1, the text media handler does not display the background color, so that the text overlay background tracks. This flag's value is 0x4000.
Text justification	A 32-bit integer that indicates how the text should be aligned. Set this field to 0 for left-justified text, to 1 for centered text, and to –1 for right-justified text.	
Background color	A 48-bit RGB color that specifies the text's background color.	
Default text box	A 64-bit rectangle that specifies an area to receive text (top, left, bottom, right). Typically this field is set to all zeros.	
Reserved	A 64-bit value that must be set to 0.	
Font number	A 16-bit value that must be set to 0.	
Font face	A 16-bit integer that indicates the font's style. Set this field to 0 for normal text. You can enable other style options by using one or more of the following bit masks:	
	0x0001	Bold
	0x0002	Italic
	0x0004	Underline
	0x0008	Outline
	0x0010	Shadow
	0x0020	Condense
	0x0040	Extend
Reserved	An 8-bit value that must be set to 0.	
Reserved	A 16-bit value that must be set to 0.	
Foreground color	A 48-bit RGB color that specifies the text's foreground color.	
Text name	A Pascal string specifying the name of the font to use to display the text.	

Text Sample Data

The format of the text data is a 16-bit length followed by the actual text. The length word specifies the number of bytes of text, not including the length word itself. Following the text, there may be one or more atoms containing additional information for drawing and searching the text.

Table 1-9 lists the currently defined text sample extensions.

Table 1-9 Text sample extensions

Text sample extension	Description
`'styl'`	Style information for the text. Allows you to override the default style in the sample description or to define more than one style for a sample. The data is a TextEdit style scrap.
`'ftab'`	Table of font names. Each table entry contains a font number (stored in a 16-bit integer) and a font name (stored in a Pascal string). This atom is required if the `'styl'` atom is present.
`'hlit'`	Highlight information. The atom data consists of two 32-bit integers. The first contains the starting offset for the highlighted text, and the second has the ending offset. Highlight samples can be in key frames or in differenced frames. When it's used in a differenced frame, the sample should contain a zero-length piece of text.
`'hclr'`	Highlight color. This atom specifies the 48-bit RGB color to use for highlighting.
`'drpo'`	Drop shadow offset. When the Display flags indicate drop shadow style, this atom can be used to override the default drop shadow placement. The data consists of two 16-bit integers. The first indicates the horizontal displacement of the drop shadow, in pixels; the second, the vertical displacement.
`'drpt'`	Drop shadow transparency. The data is a 16-bit integer between 0 and 256 indicating the degree of transparency of the drop shadow. A value of 256 makes the drop shadow completely opaque.
`'imag'`	Image font data. This atom contains two more atoms. An `'idat'` atom contains compressed image data used to draw the text when the required fonts are not available. An `'idsc'` atom contains a video sample description describing the format of the compressed image data.
`'metr'`	Image font highlighting. This atom contains metric information that governs highlighting when an `'imag'` atom is used for drawing.

Music Media

Music media is used to store note-based audio data, such as MIDI data, in QuickTime movies. It has a media type of `'musi'`.

Music Sample Description

The music sample description uses the standard sample description header, as described in "Sample Table Atoms" beginning on page 45.

The data format field in the sample description is always set to `'musi'`. The music media handler adds an additional 32-bit integer field to the sample description containing flags. Currently no flags are defined, and this field should be set to 0.

Following the Flags field, there may be appended data in the QuickTime music format. This data consists of part-to-instrument mappings in the form of general events containing note requests. One note request event should be present for each part that will be used in the sample data.

Music Sample Data

The sample data for music samples consists entirely of data in the QuickTime music format. Typically, up to 30 seconds of notes will be grouped into a single sample. For a complete description of the QuickTime music format, see your most recent QuickTime developer update documentation.

MPEG Media

MPEG media is used to store MPEG streams in QuickTime movies. It has a media type of `'MPEG'`.

MPEG Sample Description

The MPEG sample description uses the standard sample description header, as described in “Sample Table Atoms” beginning on page 45.

The Data format field in the sample description is always set to `'MPEG'`. The MPEG media handler adds no additional fields to the sample description.

MPEG Sample Data

Each sample in an MPEG media is an entire MPEG stream. This can mean that a single MPEG sample may be several hundred megabytes in size. The MPEG encoding used by QuickTime corresponds to the ISO standard, as described in ISO document CD 11172.

Sprite Media

Sprite media is used to store character-based animation data in QuickTime movies. It has a media type of `'sprt'`.

Sprite Sample Description

The base sample description uses the standard sample description header, as described in “Sample Table Atoms” beginning on page 45.

The Data format field in the sample description is always set to `'sprt'`. The base media handler adds no additional fields to the sample description.

Sprite Sample Data

All sprite samples are stored in QT atom structures. The sprite media uses both key frames and differenced frames. The key frames contain all of the sprite's image data and the initial settings for each of the sprite's properties.

A key frame always contains a shared data atom of type `'dflt'`. This atom contains data to be shared between the sprites, consisting mainly of image data and sample descriptions. The shared data atom contains a single sprite image container atom, with an atom type value of `'imct'` and an ID value of 1.

The sprite image container atom stores one or more sprite image atoms of type `'imag'`. Each sprite image atom contains a video sample description immediately followed by the sprite's compressed image data. The sprite image atoms should have ID numbers starting at 1 and counting consecutively upward.

The key frame must also contain definitions for each sprite in atoms of type `'sprt'`. Sprite atoms should have ID numbers that start at 1 and count consecutively upward. Each sprite atom contains a list of properties. Four of the properties are required: Image index, Matrix, Layer, and Visibility. Table 1-10 shows all currently defined sprite properties.

Table 1-10 Sprite properties

Property Name	Property Value	Description
Matrix	1	Specifies a transformation matrix that describes the sprite's location, scaling, and so on.
Layer	5	Contains a 16-bit integer value specifying the layer into which the sprite is to be drawn (lower layers are rendered on top of higher layers)
Visibility	4	Contains a 16-bit integer that controls the sprite's visibility: set to 1 if the sprite is visible; 0 if not
Graphics mode	6	Contains a 32-bit integer specifying the sprite's graphics mode; this value is always followed by a 48-bit RGB value
Image index	100	Contains the atom ID of the sprite's image atom

The frame difference sample differs from the key frame sample in two ways. First, the frame difference sample does not contain a shared data atom. All shared data must appear in the key frame. Second, only those sprite properties that change need to be specified. If none of a sprite's properties change in a given frame, then the sprite does not need an atom in the differenced frame.

The frame difference sample can be used in one of two ways: the frame differences can be combined, as with video key frames, to construct the current frame; or the current frame can be derived by combining only the key frame and the current frame difference.

Base Media

Base media is used to store data that is not intended to be displayed in QuickTime movies. The data may be stored for an application to retrieve, or it may be used to modify another track. It has a media type of `'gnrc'`. Other media types are often built on top of the base media. Examples of media types derived from the base media include MPEG, sprite, text, music, and time code.

Base Sample Description

The base sample description uses the standard sample description header, as described in "Sample Table Atoms" beginning on page 45.

The Data format field in the sample description is always set to `'gnrc'`. The base media handler adds no additional fields to the sample description.

Base Sample Data

The base media handler defines no data formats. Applications may put whatever data they require into the samples.

Tween Media

Tween media is used to store pairs of values to be interpolated between in QuickTime movies. These interpolated values modify the playback of other media types by using track references and track input maps. For example, a tween media can generate gradually-changing relative volume levels to cause an audio track to fade out. It has a media type of `'twen'`.

Tween Sample Description

The tween sample description uses the standard sample description header, as described in "Sample Table Atoms" beginning on page 45.

The Data format field in the sample description is always set to `'twen'`. The tween media handler adds no additional fields to the sample description.

Tween Sample Data

Tween sample data is stored in QT atom structures.

At the root level, there are one or more tween entry atoms; these atoms have an atom type value of `'twen'`. Each tween entry atom completely describes one interpolation operation. These atoms should be consecutively numbered starting at 1, using the Atom ID field.

Each tween entry atom contains several more atoms that describe how to perform the interpolation. The Atom ID field in each of these atoms must be set to 1.

- Tween start atom (atom type is `'twst'`). This atom specifies the time at which the interpolation is to start. The time is expressed in the media's time coordinate system,

relative to the start of the media sample. If this atom is not present, the starting offset is assumed to be 0.

- Tween duration atom (atom type is `'twdu'`). This atom specifies how long the interpolation is to last. The time is expressed in the media's time coordinate system. If this atom is not present, the duration is assumed to be the length of the sample.
- Tween data atom (atom type is `'twdt'`). This atom contains the actual values for the interpolation. The contents depend on the value of the tween type atom.
- Tween type atom (atom type is `'twnt'`). Describes the type of interpolation to perform. Table 1-11 shows all currently defined tween types. All tween types are currently supported using linear interpolation.

Table 1-11 Tween type values

Tween type	Value	Tween data
16-bit integer	1	Two 16-bit integers.
32-bit integer	2	Two 32-bit integers.
32-bit fixed-point	3	Two 32-bit fixed-point numbers.
Point: two 16-bit integers	4	Two points.
Rectangle: four 16-bit integers	5	Two rectangles.
QuickDraw region	6	Two matrices. The tween entry atom must contain a `'qdrg'` atom with an Atom ID value of 1. The region is transformed through the interpolated matrices.
Matrix	7	Two matrices.
RGB color: three 16-bit integers	8	Two RGB colors.
Graphics mode with RGB color	9	Two graphics modes with RGB color. Only the RGB color is interpolated. The graphics modes must be the same.

3D Media

QuickTime movies store 3D image data in a base media. This media has a media type of `'qd3d'`.

3D Sample Description

The 3D sample description uses the standard sample description header, as described in "Sample Table Atoms" beginning on page 45.

The Data format field in the sample description is always set to `'qd3d'`. The 3D media handler adds no additional fields to the sample description.

3D Sample Data

The 3D samples are stored in the 3D metafile format developed for QuickDraw 3D. Please refer to *Inside Macintosh: 3D Metafile Reference* for details.

Basic Data Types

This section describes a number of common data types that are used in QuickTime files. For information about other QuickTime data types, refer to *Inside Macintosh: QuickTime*.

Language Code Values

Some elements of a QuickTime file may be associated with a particular spoken language. To indicate the language associated with a particular object, the QuickTime file format uses language codes from the Macintosh Script Manager. Table 1-12 lists the language codes supported by QuickTime.

Table 1-12 QuickTime language code values

Language	Value	Language	Value
English	0	Georgian	52
French	1	Moldavian	53
German	2	Kirghiz	54
Italian	3	Tajiki	55
Dutch	4	Turkmen	56
Swedish	5	Mongolian	57
Spanish	6	MongolianCyr	58
Danish	7	Pashto	59
Portuguese	8	Kurdish	60
Norwegian	9	Kashmiri	61
Hebrew	10	Sindhi	62
Japanese	11	Tibetan	63
Arabic	12	Nepali	64
Finnish	13	Sanskrit	65
Greek	14	Marathi	66
Icelandic	15	Bengali	67
Maltese	16	Assamese	68

Table 1-12 QuickTime language code values (continued)

Language	Value	Language	Value
Turkish	17	Gujarati	69
Croatian	18	Punjabi	70
Traditional Chinese	19	Oriya	71
Urdu	20	Malayalam	72
Hindi	21	Kannada	73
Thai	22	Tamil	74
Korean	23	Telugu	75
Lithuanian	24	Sinhalese	76
Polish	25	Burmese	77
Hungarian	26	Khmer	78
Estonian	27	Lao	79
Lettish	28	Vietnamese	80
Latvian	28	Indonesian	81
Saamisk	29	Tagalog	82
Lappish	29	MalayRoman	83
Faeroese	30	MalayArabic	84
Farsi	31	Amharic	85
Persian	31	Tigrinya	86
Russian	32	Galla	87
Simplified Chinese	33	Oromo	87
Flemish	34	Somali	88
Irish	35	Swahili	89
Albanian	36	Ruanda	90
Romanian	37	Rundi	91
Czech	38	Chewa	92
Slovak	39	Malagasy	93
Slovenian	40	Esperanto	94
Yiddish	41	Welsh	128
Serbian	42	Basque	129
Macedonian	43	Catalan	130
Bulgarian	44	Latin	131
Ukrainian	45	Quechua	132

Table 1-12 QuickTime language code values (continued)

Language	Value	Language	Value
Byelorussian	46	Guarani	133
Uzbek	47	Aymara	134
Kazakh	48	Tatar	135
Azerbaijani	49	Uighur	136
AzerbaijanAr	50	Dzongkha	137
Armenian	51	JavaneseRom	138
Georgian	52		

Calendar Date and Time Values

QuickTime movies store date and time information in Macintosh date format: a 32-bit value indicating the number of seconds that have passed since midnight, January 1, 1904.

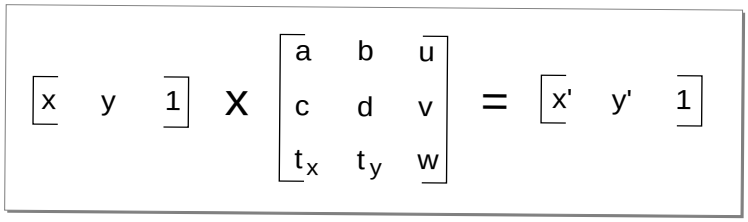
Matrices

QuickTime files use matrices to describe spatial information about many objects, such as tracks within a movie.

A transformation matrix defines how to map points from one coordinate space into another coordinate space. By modifying the contents of a transformation matrix, you can perform several standard graphics display operations, including translation, rotation, and scaling. The matrix used to accomplish two-dimensional transformations is described mathematically by a 3-by-3 matrix.

All values in the matrix are 32-bit fixed-point numbers divided as 16.16, except for the {u, v, w} column that contains 32-bit fixed-point numbers divided as 2.30. Figure 1-44 depicts how QuickTime uses matrices to transform displayed objects.

Figure 1-44 How display matrices are used in QuickTime



Graphics Modes

QuickTime files use graphics modes to describe how one video or graphics layer should be combined with the layers beneath it. Graphics modes are also known as transfer modes. Some graphics modes require a color to be specified for certain operations, such as blending to determine the blend level. QuickTime uses the graphics modes defined by QuickDraw. For information on QuickDraw graphics modes, see *Inside Macintosh: Imaging*.

The most common graphics modes are `srcCopy` (0x00) and `ditherCopy` (0x040), which simply indicate that the image should not blend with the image behind it, but overwrite it. QuickTime also defines several new graphics modes.

Table 1-13 lists the new graphics modes supported by QuickTime.

Table 1-13 QuickTime graphics modes

Graphics mode value	Description
0x0100	Straight alpha
0x0101	Alpha pre-multiplied with white
0x0102	Alpha pre-multiplied with black

RGB Colors

Many atoms in the QuickTime file format contain RGB color values. These are usually stored as three consecutive unsigned 16-bit integers in the following order: red, green, blue.

Examples

This section contains a number of examples that help you pull together all of the material in this book by examining the atom structure that results from a number of different scenarios. This section is divided into the following topics:

- "Creating Video Tracks at 30 Frames-per-second" discusses creating 30 fps video
- "Creating Video Tracks at 29.97 Frames-per-second" describes creating 29.97 fps video
- "Creating Audio Tracks at 44.1Khz" provides an example of creating an audio track
- "Creating a Time Code Track for 29.97 FPS Video" presents a time code track example
- "Playing With Edit Lists" discusses how to interpret edit list data
- "Interleaving Movie Data" shows how a movie's tracks are interleaved in the movie data file

- "Referencing Two Data Files With a Single Track" shows how track data may reside in more than one file

Creating Video Tracks at 30 Frames-per-second

The duration of a video frame is stored in the time-to-sample atom contained within a sample table atom. This duration cannot be interpreted without the media's time scale, which defines the units-per-second for the duration. In this example, each frame has the same duration, so the time-to-sample atom has one entry that applies to all video frames in the media.

As long as the ratio between frame duration and media time scale remains 1:30, any combination of values can be used for the duration and time scale. The larger the time scale the shorter the maximum duration. Since a movie defaults to a time scale of 600, this is a good number to use. It is also the least common multiple for 24, 25, and 30, making it handy for much of the math you are likely to encounter when making a movie.

The movie time scale is independent of the media time scale. Since you want to avoid movie edits that don't land on frame boundaries, it is a good idea to keep the movie time scale and the media time scale the same, or the movie time scale should be an even multiple of the media time scale. The movie time scale is stored in the movie header atom.

With a time scale of 600 in the media header atom, the time-to-sample atom would contain the following data values:

Atom size	24
Atom type	`'stts'`
Version/Flags	0
Number of entries	1
Sample count	n
Sample duration	20

Creating Video Tracks at 29.97 Frames-per-second

NTSC color video is not 30 frames-per-second (fps), but actually 29.97 fps. The previous example showed how the media time scale and the duration of the frames specify the video's frame rate. By setting the media's time scale to 2997 units per second and setting the frame durations to 100 units each, the effective rate is 29.97 fps exactly.

In this situation, it is also a good idea to set the movie time scale to 2997 in order to avoid movie edits that don't land on frame boundaries. The movie's time scale is stored in the movie header atom.

With a time scale of 2997 in the media header atom, the time-to-sample atom would contain the following data values:

Atom size	24
Atom type	`'stts'`
Version/Flags	0
Number of entries	1
Sample count	n
Sample duration	100

Creating Audio Tracks at 44.1Khz

The duration of an audio sample is stored in the time-to-sample atom contained in a sample table atom. This duration cannot be interpreted without the media's time scale, which defines the units-per-second for the duration. With audio, the duration of each audio sample is typically 1, so the time-to-sample atom has one entry that applies to all audio samples.

With a time scale of 44100 in the media header atom, the time-to-sample atom would contain the following data values:

Atom size	24
Atom type	`'stts'`
Version/Flags	0
Number of entries	1
Sample count	n
Sample duration	1

This atom does not indicate whether the audio is stereo or mono or whether it contains 8-bit or 16-bit samples. That information is stored in the sound sample description atom, which is contained in the sample table atom.

Creating a Time Code Track for 29.97 FPS Video

A time code track specifies time code information for other tracks. The time code keeps track of the time codes of the original source of the video and audio. After a movie has been edited, the time code can be extracted to determine the source tape and the time codes of the frames.

It is important that the time code track has the same time scale as the video track. Otherwise, the time code will not tick at the exact same time as the video track.

For each contiguous source tape segment, there is a single time code sample that specifies the time code value corresponding to the start of the segment. From this sample, the time code value can be determined for any point in the segment.

The sample description for a time code track specifies the time code system being used (for example, 30 fps drop-frame) and the source information. Each sample is a time code value.

Since the time code media handler is a derived from the base media handler, the media information atom starts with a generic media header atom. The time code atoms would contain the following data values:

Atom size	77		
Atom type	`'gmhd'`		
	Atom size	69	
	Atom type	`'gmin'`	
	Version/Flags	0	
	Graphics mode	0x0040	
	Opcolor (red)	0x8000	
	Opcolor (green)	0x8000	
	Opcolor (blue)	0x8000	
	Balance	0	
	Reserved	0	
	Atom size	45	
	Atom type	`'tmcd'`	
		Atom size	37
		Atom type	`'tcmi'`
		Version/Flags	0
		Text font	0 (system font)
		Text face	0 (plain)
		Text size	12
		Text color (red)	0
		Text color (green)	0
		Text color (blue)	0
		Background color (red)	0
		Background color (green)	0
		Background color (blue)	0
		Font name	`"\pChicago"` (Pascal string)

The sample table atom contains all the standard sample atoms and has the following data values:

Atom size	174		
Atom type	`'stbl'` (sample table)		
	Atom size	74	
	Atom type	`'stsd'` (sample description)	
	Version/Flags	0	
	Number of entries	1	
	Sample description size [1]	58	
	Data format [1]	`'tmcd'`	
	Reserved [1]	0	
	Data reference index [1]	1	
	Flags [1]	0	
	Flags (time code) [1]	7 (drop frame + 24 hour + negative times ok)	
	Time scale[1]	2997	
	Frame duration[1]	100	
	Number of frames[1]	20	
		Atom size	24
		Atom type	`'name'`
		String length	12
		Language code	0 (English)
		Name	“my tape name”
	Atom size	24	
	Atom type	`'stts'` (time-to-sample)	
	Version/Flags	0	
	Number of entries	1	
	Sample count[1]	1	
	Sample duration[1]	1	
	Atom size	28	
	Atom type	`'stsc'` (sample-to-chunk)	
	Version/Flags	0	
	Number of entries	1	
	First chunk[1]	1	
	Samples per chunk[1]	1	

Sample description ID[1]	1
Atom size	20
Atom type	`'stsz'` (sample size)
Version/Flags	0
Sample size	4
Number of entries	1
Atom size	20
Atom type	`'stco'` (chunk offset)
Version/Flags	0
Number of entries	1
Offset[1]	(offset into file of chunk 1)

In the example, let's assume that the segment's beginning time code is 1:15:32.4 (1 hour, 15 minutes, 32 seconds, and 4 frames). This time would be expressed in the data file as 0x010F2004 (0x01 = 1 hour; 0x0F = 15 minutes; 0x20 = 32 seconds; 0x04 = 4 frames).

The video and audio tracks must contain a track reference atom to indicate that they reference this time code track. The track reference is the same for both and is contained in the track atom (at the same level as the track header and media atoms).

This track reference would contain the following data values:

Atom size	12
Atom type	`'tref'`
Reference type	`'tmcd'`
Track ID of referenced track (time code track)	3

In this example, the video and sound tracks are tracks 1 and 2. The time code track is track 3.

Playing With Edit Lists

A segment of a movie can be repeated without duplicating media data by using edit lists. Suppose you have a single-track movie whose media time scale is 100 and track duration is 1000 (10 seconds). For this example the movie's time scale is 600. If there are no edits in the movie, the edit atom would contain the following data values:

Atom size	36	
Atom type	`'edts'`	
	Atom size	28
	Atom type	`'elst'`
	Version/Flags	0
	Number of entries	1
	Track duration	6000 (10 seconds)
	Media time	0
	Media rate	1.0

Because this is a single-track move, the track's duration in the track header atom is 6000, and the movie's duration in the movie header atom is 6000.

If you change the track to play the media from time 0 to time 2 seconds, and then play the media from time 0 to time 10 seconds, the edit atom would now contain these data values:

Atom size	48	
Atom type	`'edts'`	
	Atom size	40
	Atom type	`'elst'`
	Version/Flags	0
	Number of entries	2
	Track duration[1]	1200 (2 seconds)
	Media time[1]	0
	Media rate[1]	1.0
	Track duration[2]	6000 (10 seconds)
	Media time[2]	0
	Media rate[2]	1.0

Because the track is now 2 seconds longer, the track's duration in the track header atom must now be 7200, and the movie's duration in the movie header atom must also be 7200.

Currently, the media plays from time 0 to time 2, then plays from time 0 to time 10. If you take that repeated segment at the beginning (time 0 to time 2) and play it at double speed to maintain the original duration, the edit atom would now contain the following values:

Atom size	60	
Atom type	`'edts'`	
	Atom size	52
	Atom type	`'elst'`
	Version/Flags	0
	Number of entries	3
	Track duration[1]	600 (1 seconds)
	Media time[1]	0
	Media rate[1]	2.0
	Track duration[2]	600 (1 second)
	Media time[2]	0
	Media rate[2]	2.0
	Track duration[3]	4800 (8 seconds)
	Media time[3]	200
	Media rate[3]	1.0

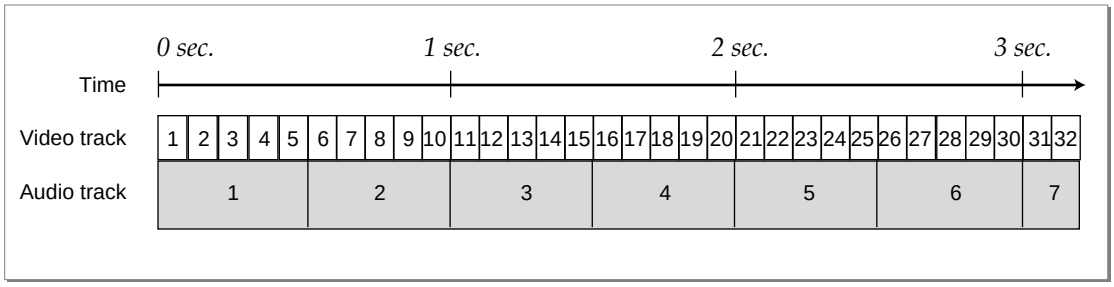
Because the track is now back to its original duration of 10 seconds, its duration in the track header atom is 6000 and the movie's duration in the movie header atom is 6000.

Interleaving Movie Data

In order to get optimal movie playback, you must create the movie with interleaved data. The data for the movie is placed on disk in time order so the video, sound, and other data for a particular time in the movie are close together in the file. This means that you will have to intersperse the data from different tracks. To illustrate this, consider a movie with a single video and a single audio track.

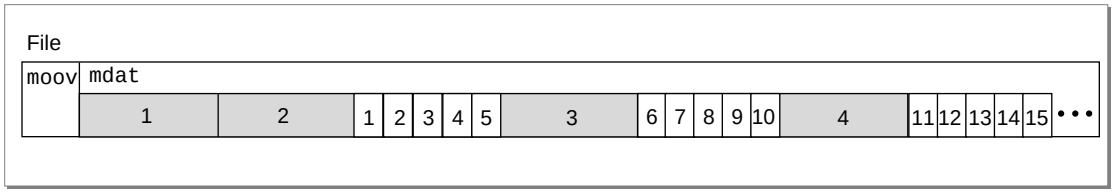
Figure 1-45 shows how the movie data was collected, and how the data would need to be played back for proper synchronization. In this example, the video data is recorded at 10 frames per second and the audio data is grouped into 1/2-second chunks.

Figure 1-45 Noninterleaved movie data



After the data has been interleaved on the disk, the movie data atom contains movie data in the order shown in Figure 1-46.

Figure 1-46 Interleaved movie data



In this example, the file begins with the movie atom (`'moov'`), followed by the movie data atom (`'mdat'`). In order to overcome any latencies in sound playback, at least one second of sound data is placed at the beginning of the interleaved data. This means that the sound and video data are offset from each other in the file by one second.

Referencing Two Data Files With a Single Track

The data reference index to be used for a given media sample is stored within that sample's sample description. Therefore, a track must contain multiple sample descriptions in order for that track to reference multiple data files. A different sample description must be used whenever the data file changes or whenever the format of the data changes. The sample-to-chunk atom determines which sample description to use for a sample.

The sample description atom would contain the following data values:

Atom size	…	
Atom type	`'stsd'`	
Version/Flags	0	
Number of entries	2	
	Sample description size[1]	…
	Data format	`'tmcd'`
	Reserved	0
	Data reference index	1
	(sample data)	…
	Sample description size[1]	…
	Data format	`'tmcd'`
	Reserved	0
	Data reference index	2
	(sample data)	…

If there were only 1 sample per chunk and the first 10 samples were extracted from sample description 2 and the next 30 samples were extracted from sample description 1, the sample-to-chunk atom would contain the following data values:

Atom size	40
Atom type	`'stsc'`
Version/Flags	0
Number of entries	2
First chunk[1]	1
Samples per chunk[1]	1
Sample description ID[1]	2
First chunk[2]	11
Samples per chunk[2]	1
Sample description ID[2]	1

The data reference atom would contain the following data values:

Atom size	…	
Atom type	`'dinf'`	
	Atom size	…
	Atom type	`'dref'`
	Version/Flags	0
	Number of entries	2
	Size[1]	…
	Type[1]	`'alis'`
	Version[1]	0
	Flags[1]	0 (not self referenced)
	Data reference[1]	[alias pointing to file #1]
	Size[2]	…
	Type[2]	`'alis'`
	Version[2]	0
	Flags[2]	0 (not self referenced)
	Data reference[2]	[alias pointing to file #2]

Index

Symbols

'©cpy' user data type 58
'©day' user data type 58
'©dir' user data type 58
'©ed1' to '©ed9' user data types 58
'©fmt' user data type 58
'©inf' user data type 58
'©prd' user data type 58
'©prf' user data type 58
'©req' user data type 58
'©src' user data type 58
'©wrt' user data type 58
µLaw sound 65

Numerals

16-bit sound, uncompressed 65
3D media 74
3D media sample data 75
3D sample description 74
8-bit sound, uncompressed 65

A

'AllF' user data type 58
anti-aliasing text 69

B

balance, media sound 38, 40
base media 73
base media info atoms 39 to 40
base media information atoms 34, 38 to 39
base media information header atoms 39 to 40
base media sample data 73
base sample description 73

C

channels, sound 64
child count, QT atoms 5
chunk offset atoms 46, 55 to 56, 57
chunks 44
'clip' atoms 19
clipping atoms 11, 19
clipping region atoms 20
color table, media 61
color table, movie 15
color table atoms 11, 14 to 15
compressed matte atoms 21 to 22
compression formats, sound 64
container atoms 2
copyright statement, user data type for 58
creation time, media 32
creation time, movie 13
creation time, track 18
'crgn' atoms 20
'ctab' atoms 14

D

data information atoms 35, 37, 41 to 44
data reference atoms 43 to 44
depth, pixel 60
'dflt' atoms 72
dimensions, track 19
'dinf' atoms 41
'dref' atoms 43
drop shadow, text 69
'drpo' atoms 70
'drpt' atoms 70
duration, media 32
duration, movie 14
duration, track 18

E

edit atoms 15, 22 to 23
edit list, track 22
edit list atoms 23 to 24
'edts' atoms 22
'elst' atoms 23
extension, file 6

F

file types and extensions 6
font, text 69
'free' atoms 7
free space atoms 7
'ftab' atoms 70

G

gamma level, media 61
'gmhd' atoms 39
'gmin' atoms 39
'gnrc' media type 73
graphics mode, media 36, 40
graphics modes 78

H

handler reference atoms 30, 33 to 34, 35, 37
'hclr' atoms 70
'hdlr' atoms 33
header layout, atom 3
'hlit' atoms 70
Huffman table 61, 63

I

ID, QT atoms 5
'idat' atoms 70
'idsc' atoms 70
'imag' atoms 70, 72
image compression formats 59
image resolution 60
image size, media 60
'imap' atoms 27
IMA sound 65
'imct' 72
'imct' atoms 72
' in' atoms 29

J

JPEG video 62
justification, text 69

K

key frames 50
key frame sample, finding 57
'kmat' atoms 21

L

language, media 32
language codes 75 to 77
layer, track 18
leaf atoms 2
limitations of atoms 3
'load' atoms 24
looping, user data type for 58
'LOOP' user data type 58

M

MACE sound 65
matrix, movie display 14
matrix, track display 18
matrix contents 77
'matt' atoms 20
matte, track 20
'mdat' atoms 7
'mdhd' atoms 31
'mdia' atoms 30
media atoms 15, 30 to 34
media data atoms 59 to 75
media handler components 33
media header atoms 30, 31 to 33
media information atoms 30, 34 to 40
  time code 66
'metr' atoms 70
'minf' atoms 34
modification time, media 32
modification time, movie 14
modification time, track 18
'moov' atoms 8
Motion JPEG video data 62 to 63
movie atoms 8 to 59
movie data atoms 7
movie header atoms 11, 12 to 14
movies
  color table 15
  creation dates, user data type for 58
  creation time 13
  credits in, user data type for 58
  current selection 14
  default window location, user data type for 58

director names, user data type for 58
duration 14
edit dates and descriptions, user data type for 58
formats, user data type for 58
hardware requirements, user data type for 58
information about, user data type for 58
looping type, user data type for 58
matrix 14
modification time 14
object, user data type for 58
performers, user data type for 58
playback rate 14
playing all frames, user data type for 58
play selection only, user data type for 58
poster 14
preview 14
producer, user data type for 58
software requirements, user data type for 58
sound volume 14
time scale 14
track ID, next 14
writers of, user data type for 58
movies and QuickTime files 1
MPEG media 71
MPEG media sample data 71
'MPEG' media type 71
MPEG sample description 71
MPEG video 64
music media 70
music media sample data 71
music sample description 70
'musi' media type 70
'mvhd' atoms 12

N

'name' user data type 58
no lean ahead flag, media 36

O

'obid' atoms 30
object name, user data type for 58
order of atoms 2, 5, 7

P

pixel depth 60
playback hints, track 25
'pnot' atoms 7
poster, movie 14
preload information, track 25
preview, movie 14
preview atoms 7 to 8

Q

'qd3d' media type 74
'qdrg' atoms 74
QT atoms 3 to 5
quality, media playback 33
quality, video 60
quantization table 61, 62, 63
QuickTime 1.0 media compatibility 36
QuickTime files and movies 1

R

rate, movie 14
resolution, image 60
RGB video, uncompressed 61

S

sample, finding 56
sample atoms 44 to 56
using 56 to 57
sample data
3D media 75
base media 73
MPEG media 71
music media 71
sprite media 72
text media 69
tween media 73
sample data formats, sound 65
sample data formats, video 61 to 64
sample description atoms 46, 47 to 48, 53
sample descriptions
3D 74
base 73
MPEG 71
music 70
sound 64 to 65
sprite 71
text 68
time code 66
tween 73

video 59 to 61
sample size, sound 65
sample size atoms 46, 53 to 55, 56, 57
sample table atoms 35, 37, 45 to 46
sample-to-chunk atoms 46, 47, 51 to 53, 56, 57
'Sel0' user data type 58
selection, movie 14
selection, playing 58
shadow sync atoms 46
size, atom 3
'skip' atoms 7
'smhd' atoms 37
sound balance, media 38, 40
sound compression formats 64
sound media 64
sound media handler 34
sound media information atoms 34 to 37
sound media information header atoms 38
sound sample data formats 65
sound sample description 64 to 65
sound sample size 65
sound volume, movie 14
'soun' media type 64
sprite display, controlling 72
sprite media 71
sprite media sample data 72
sprite sample description 71
'sprt' atoms 72
'sprt' media type 71
'stbl' atoms 45
'stco' atoms 55
'stsc' atoms 51
'stsd' atoms 47
'stss' atoms 50
'stsz' atoms 53
'stts' atoms 48
'styl' atoms 70
sync sample atoms 46, 50 to 51, 57

T

text, anti-aliasing 69
text color, controlling 68
text font 69
text justification 69
text media 68
text media display, controlling 68
text media drop shadow 69
text media sample data 69
'text' media type 68
text sample description 68
text scrolling, controlling 68
time code media 65
time code media information atoms 66
time code sample data 67
time code sample description 66
time scale, media 32
time scale, movie 14
time-to-sample atoms 46, 48 to 50, 56, 57
'tkhd' atoms 17
'tmcd' media type 65
track atoms 11, 15 to 30
track clipping atoms 15
track header atoms 15, 17 to 19
track ID, next in a movie 14
track input map atoms 15, 27 to 30
track load settings atoms 24 to 25
track matte atoms 15, 20 to 21
track reference atoms 15, 26 to 27
tracks 18
creation time 18
dimensions 19
double-buffered I/O 25
duration 18
edit list 22
enabled flag 18
ID value 18
layer 18
matrix 18
matte 20
modification time 18
playback hints 25
preload information 25
references between tracks 26
secondary data sources 27
sound volume 18
used in poster flag 18
'trak' atoms 15
transfer mode, media 36, 40
'tref' atoms 26
'twdt' atoms 74
'twdu' atoms 74
tween data atoms 74
tween duration atoms 74
tween entry atoms 73
tween media 73
tween media sample data 73
tween sample description 73
tween start atoms 73
tween type atoms 74
'twen' atoms 73
'twen' media type 73
'twnt' atoms 74
'twst' atoms 73
' ty' atoms 29
type, atom 3
type, QT atoms 5
types, file 6

U

'udta' atoms 57
used in movie flag 18
used in preview flag 18
user data atoms 11, 15, 30, 57 to 59
user data types 58

V

'vide' media type 59
Video 34, 35
video compression formats 59
video media 59
video media handler 34
video media information atoms 34 to 35
video media information header atoms 35 to 36
video quality 60
video sample data formats 61 to 64
video sample description 59 to 61
'vmhd' atoms 35
volume, movie sound 14
volume, track sound 18

W, X

'WLOC' user data type 58

Y, Z

YUV video, uncompressed 62

This Apple manual was written, edited, and composed on a desktop publishing system using Apple Macintosh computers and FrameMaker software. Proof pages and final pages were created on an Apple LaserWriter Pro printer. Line art was created using Adobe™ Illustrator and Adobe Photoshop. PostScript™, the page-description language for the LaserWriter, was developed by Adobe Systems Incorporated.

Text type is Palatino® and display type is Helvetica®. Bullets are ITC Zapf Dingbats®. Some elements, such as program listings, are set in Apple Courier.

LEAD WRITER
Linda Kyrnitszke

WRITER
Doug Engfer

DEVELOPMENTAL EDITOR
Wendy Krafft

ILLUSTRATORS
Deb Dennis, Sandee Karr

PRODUCTION EDITOR
Gerri Gray

Special thanks to Peter Hoddie