iTunes Metadata Format Specification

2008-04-16

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

Contents

Figures and Tables

Introduction

Apple's iTunes Metadata Format is a format for storing and transporting metadata in iTunes audio and video files. It simplifies the management and manipulation of such file-specific data within both protected and unprotected media files: `.m4a` (unprotected audio), `.m4p` (protected audio), `.m4v` (protected or unprotected video), and `.mp4` (unprotected audio and/or video). This document describes the metadata and format for iTunes, version 6 and later.

This document is intended for developers who create or modify files for use with iTunes in order to include metadata such as artist name, copyright, and so on.

Organization of This Document

This document contains a single chapter, "The iTunes Metadata Format" (page 9), which provides a brief overview of the metadata associated with iTunes music and video files and details the structure and use of specific metadata currently supported for use with iTunes.

See Also

The following documents provide additional resources:

- ISO/IEC standards 14496-12:2005 and 15444-12, freely available from the International Organization for Standardization (ISO), which specify the structure and uses of the ISO base media file format.
- ISO/IEC standard 14496-14, available from the International Standards Organization (ISO), which specifies the MPEG-4 file format.

The iTunes Metadata Format

This document defines the format of metadata stored in Apple iTunes audio and video files.

LICENSE: The Apple file format documentation (“Apple Specification”) contained in this publication is supplied to you by Apple Inc. (“Apple”) in consideration of your agreement to the following terms, and your use of this Apple Specification constitutes acceptance of these terms. If you do not agree with these terms, please do not use this Apple Specification.

In consideration of your agreement to abide by the following terms, and subject to the terms, Apple grants you a personal, non-exclusive, limited license, under Apple’s copyrights in this Apple Specification, to implement the Apple Specification in your products solely for the purpose of enabling your products to write, read, and/or display meta-data information in files that are compatible with the ISO/IEC 14496-14 specification using the Apple Specification. You may not make or distribute copies of the Apple Specification, or electronically transfer the Apple Specification outside your company.

Neither the name, trademarks, service marks or logos of Apple may be used to endorse or promote products implementing the Apple Specification without specific prior written permission from Apple.

Except as expressly stated in this notice, no other rights or licenses, express or implied, are granted by Apple herein, including but not limited to any patent rights. Apple retains all right, title and interest in the Apple Specification.

You understand that Apple may amend, modify, change and cease distribution of the Apple Specification at any time. You understand that this license does not entitle you to any upgrades, updates or future versions of the Apple Specification under this license.

You agree to indemnify and hold Apple harmless from any third party claim, loss or damage (including attorney’s fees) related to your use of the Specification or your implementation of the Apple Specification in your products.

The Apple Specification is provided on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND FROM APPLE. YOU ASSUME ALL RISKS THAT THE APPLE SPECIFICATION IS SUITABLE OR ACCURATE FOR YOUR NEEDS AND YOUR USE OF THE SPECIFICATION IS AT YOUR OWN RISK AND DISCRETION. APPLE MAKES NO WARRANTIES, EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, REGARDING THE APPLE SPECIFICATION OR ITS USE AND OPERATION ALONE OR IN COMBINATION WITH YOUR PRODUCTS.

IN NO EVENT SHALL APPLE BE LIABLE FOR ANY SPECIAL, INDIRECT, INCIDENTAL OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) ARISING IN ANY WAY OUT OF THIS LICENSE, INCLUDING THE USE, REPRODUCTION, IMPLEMENTATION AND/OR DISTRIBUTION OF THE APPLE SPECIFICATION, HOWEVER CAUSED AND WHETHER UNDER THEORY OF CONTRACT, TORT (INCLUDING NEGLIGENCE), STRICT LIABILITY OR OTHERWISE, EVEN IF APPLE HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Overview

Apple's iTunes metadata format is a flexible structure for storing and manipulating metadata (information such as album titles, artist names, and cover art that is ancillary to the music and video data) in an iTunes audio or video file. Metadata is used to protect, organize, and select music and video files, and can be used to enrich the user experience in many ways.

Apple-proprietary data, such as that used for protection, is not described in this document. Fields used by Apple for proprietary data are marked as reserved, and documented "must be set to zero." If you are creating a new file, set these fields to zero. If you are modifying an iTunes-generated file to add metadata, do not modify fields documented as reserved.

Important: Use of the information in this specification is intended for Apple Developer Center members, and is subject to the terms and conditions of your Apple Developer Center membership agreement as well as the license set forth on the previous page of this specification. Where there are inconsistencies the license on the previous page will govern.

The iTunes audio and video files covered by this specification normally have the extension `.m4a` (unprotected audio), `.m4p` (protected audio), `.m4v` (unprotected or unprotected video), or `.mp4` (unprotected audio or video). The format and use of metadata is identical for files of audio and video, protected and unprotected formats. These file formats are all based on the *ISO Base Media File Format* specification (defined by the ISO/IEC 14496-12:2005 and 15444-12 standards) and the *MPEG-4 File Format* specification (defined by the ISO/IEC 14496-14 standard).

Note: The *ISO Base Media File Format* specification (ISO/IEC 14496-12:2005) is publicly available for free download in PDF form.

The first amendment to the ISO file format introduces the metadata atom (or *box*) with type `'meta'`. iTunes uses a predecessor of this concept. In particular:

- The metadata atom `'meta'` in iTunes is stored within a user data atom `'udta'`, whereas in the ISO amendment, it is an alternative structure.
- iTunes metadata is stored in a proprietary binary format, as detailed in this document.

In the binary format, unlike in traditional user data atoms in MP4 (or QuickTime) files, there is explicit provision for

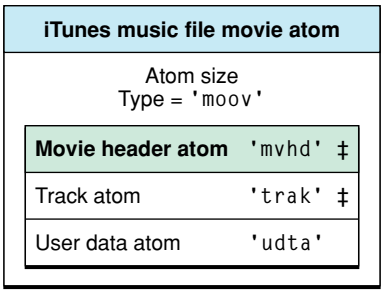
- standard tags and extension tags (also known as *meanings)*
- explicit data type information

- a reserved field for localization

Structure

An iTunes music file consists of a single track of AAC music. An iTunes video file contains an AAC music track and an MPEG-4 video track. As shown in the figure below, all metadata is stored in a user data atom inside the movie `'moov'` atom. Inside the user data atom is a single metadata atom `'meta'`.

Figure 1-1 The movie atom



‡ Required atom

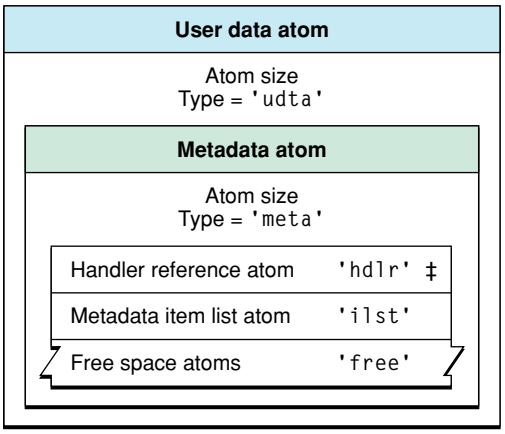
All fields in the structures described in this document are stored in big-endian format, as is normal for both QuickTime and ISO Media Files.

The Metadata Atom

The overall container for rich metadata is the `'meta'` atom. Within that atom the following atoms may occur:

- a handler atom `'hdlr'` (required)
- an item list atom `'ilst'`
- free space atoms `'free'`

Figure 1-2 The user data atom and metadata atom



‡ Required atom

To permit rewriting metadata without rewriting the entire file, free space atoms may occur within the metadata atom, in any position. They may not occur within an item or between items in the item list. This restriction avoids the risk of confusing the free space with a meaning string of `free` or a data atom value of `free`.

Unrecognized atoms should be ignored.

```
aligned(8) class MetaDataAtom extends FullAtom('meta', version, flags) {
}
```

Handler Atom

To accommodate a variety of metadata structure formats, the metadata atom is required to contain a handler reference atom `'hdlr'` specifying the handler to be used to interpret the metadata, and thereby indicating the structure of the metadata atom contents. All other contained atoms are specific to the format indicated.

This document describes the metadata structure indicated by the handler of type `'mdir'`

```
aligned(8) class HandlerAtom extends FullAtom('hdlr', version = 0, 0) {
 unsigned int(32) reserved = 0;
 unsigned int(32) handler-type;
 const unsigned int(32)[3] reserved = 0;
 string name;
}
```

Table 1-1 Handler atom fields

Field	Description
`handler-type`	An integer indicating the type of metadata atom. This document defines the structure indicated by the handler-type of `'mdir'`.
`name`	A null-terminated string of UTF-8 characters specifying a human-readable metadata type name (such as `iTunes metadata`) for debugging and inspection purposes. The string may be empty (a single byte with value 0).

The Meaning or Tag

The *tag* or *meaning* of a metadata item provides a name-space to identify what the item represents: a copyright notice, the performer's name, and so on. In user data, the tag is one of a fixed set of four-character codes. In addition to the four-character codes, this scheme also supports names in reverse-ordered ICANN domain-name format (for example, `com.apple.quicktime.windowlocation`). This feature provides an extensible syntax for vendor data or for use by other organizations or standards bodies.

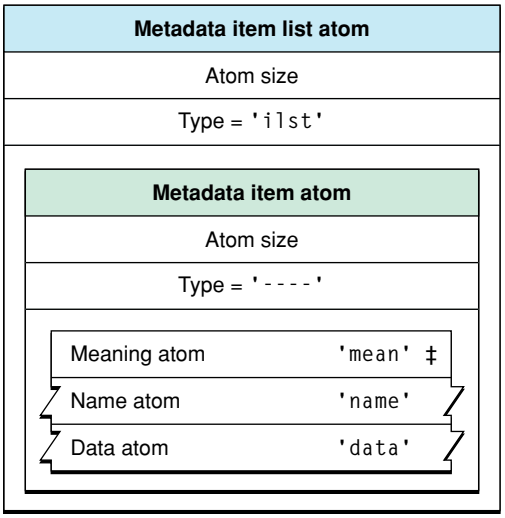
For details of the available tags, including cross-references to their MP3 ID3v4 counterpart tags, see "Defined tags" (page 18).

Item list

The metadata items are formatted as a list of items. The item list atom `'ilst'` contains a number of metadata items, each of which is an atom.

```
aligned(8) class MetaItemsAtom extends Atom('ilst') {
 MetaItemAtom item[]; // to fill atom
}
```

Figure 1-3 The metadata item list atom



‡ Required atom

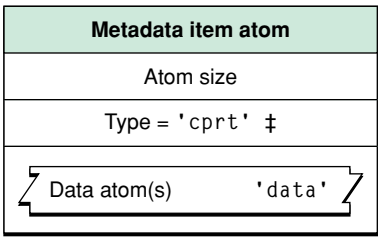
Metadata Item

Each metadata item is primarily identified by its meaning, as with today's user data items. There are two forms of the item:

- if the atom type is four dashes `'----'` then the meaning is identified by a contained meaning atom, possibly extended by a name atom, as shown in the figure above;

- otherwise, the item meaning is identified by the atom name (for example, `'cprt'` for a copyright notice, as shown in the figure below). There is a pre-defined set of these names.

Figure 1-4 The metadata item atom



‡ Or other predefined type

The following atoms may occur within a metadata item:

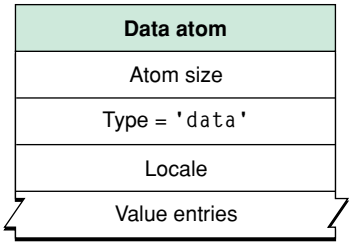
- the meaning atom `'mean'` if the metadata item atom type is `'----'`
- an optional name atom `'name'`, extending the meaning
- metadata item values

The value of the metadata item is currently expressed only as immediate in-line data, in a data atom `'data'`. All data atoms start with two common fields: a type indicator, and a locale indicator. Each of these fields is four bytes long. There may be multiple value entries; see below for the required ordering.

```
aligned(8) class MetaItemAtom extends Atom(meaning) {
 if (meaning='----') {
  MeaningAtom extended_meaning;
  NameAtom the_name; // optional
 }
 DataAtom the_value; // 1 or more of these
}

aligned(8) class DataAtom extends Atom('data') {
 type_indicator  the_type;
 unsigned int(32) the_locale = 0; // reserved to be 0
 unsigned int(8) value[]; // to fill the atom
}
```

Figure 1-5 The data atom



Type Indicator

The type indicator is formed of four bytes. The first two bytes are reserved and must be set to zero. The third byte identifies the type set, or *type space*, from which the type code is drawn. The fourth byte is the type code within that type space. All indexes start with 1.

Currently the only type set defined is the set of basic types with type set identifier 0 (zero), described in Table 1-2 (page 16).

```
aligned(8) class TypeIndicator {
 unsigned int(16) reserved = 0;
 unsigned int(8) type_set_identifier = 0;
 unsigned int(8) type_code;
}
```

To specify HTML, for example, the type set identifier would be 0, indicating the basic set of types detailed in Table 1-2 (page 16), and the type code would be 6.

Locale Indicator

The locale indicator is currently unused and must have the value 0 (zero) in files written to this specification. When such files are read, items with a non-zero locale indicator should be skipped and ignored.

Data Ordering

Multiple data atoms for the same tag can be used to provide values for variant representations of the same information. For example, an artist name might be supplied in three ways:

- as a large JPEG of his signature
- as a smaller thumbnail JPEG of his signature
- as text

Applications can choose the variation most appropriate to their needs.

In each item, variant values should be ordered from the most specific to the most general. An application might stop searching for a value upon finding a variant that it can display, or it might present all the variants and allow the user to choose one dynamically.

An application can discard variant values that are undesirable or unusable (for example, those of an irrelevant type or an inappropriate size). The resultant set of metadata consists of values that are capable of presentation. If all the tag values in a particular item are discarded, the entire item should be discarded. This can happen, for example, in the case of cover art for which only graphical data is available and is therefore irrelevant to a terminal with no graphics display capability.

Meaning

The meaning atom `'mean'` contains a name, formatted as a string of UTF-8 characters to fill the atom. This meaning name is required in a metadata item of type `'----'` and must not occur in other metadata items. The format of the name is a reverse-ordered domain name (from most general to most specific) such as com.apple.quicktime.windowlocation. The maximum length of a meaning name string may be limited in specific environments.

If a name atom also occurs, the name value is appended to the meaning, after a dot. For example, an atom with a meaning of `com.apple.iTunes` and a name value of `color` is identical to a meaning of `com.apple.iTunes.color`. For such names, iTunes currently uses and supports only this separated format.

```
aligned(8) class MeaningAtom extends FullAtom('mean', version, flags) {
 unsigned int(8) meaning-string[]; // to fill the atom
}

aligned(8) class NameAtom extends FullAtom('name', version, flags) {
 unsigned int(8) name[]; // to fill the atom
}
```

Supported Types of Data

This section describes in detail the types of data supported in the iTunes metadata format.

Basic Types

The common basic data types are grouped into a set with the type set identifier 0 (zero). Table 1-2 below lists each basic type with its type code. Some of these types are further defined in later sections. All undocumented codes are reserved.

Table 1-2 Basic data types (Set 0)

Code	Type	Notes
0	implicit	for use with tags for which no type needs to be indicated because only one type is allowed
1	UTF-8	without any count or null terminator
2	UTF-16	also known as UTF-16BE
3	S/JIS	deprecated unless it is needed for special Japanese characters
6	HTML	the HTML file header specifies which HTML version
7	XML	the XML header must identify the DTD or schemas
8	UUID	also known as GUID; stored as 16 bytes in binary (valid as an ID)

Code	Type	Notes
9	ISRC	stored as UTF-8 text (valid as an ID)
10	MI3P	stored as UTF-8 text (valid as an ID)
12	GIF	(deprecated) a GIF image
13	JPEG	in a JFIF wrapper
14	PNG	in a PNG wrapper
15	URL	absolute, in UTF-8 characters
16	duration	in milliseconds, a 32-bit integer
17	date/time	in UTC, counting seconds since midnight on 1 January, 1904; 32 or 64 bits
18	Genres	a list of values from the enumerated set (see "Genre" (page 18))
21	Integer	A signed big-endian integer in 1,2,3,4 or 8 bytes
24	RIAA-PA	RIAA Parental advisory; -1=no, 1=yes, 0=unspecified. 8-bit integer
25	UPC	Universal Product Code, in text UTF-8 format (valid as an ID)
27	BMP	Windows bitmap format graphics

Specialized Types

Track Numbers

Each file includes zero or more track number declarations, which identify its position within any playlist that could be reconstructed from the files.

Table 1-3 Track number types

Name	Length	Description
Reserved1	1 byte	Must be set to 0.
Reserved2	1 byte	Length. Must be set to 0.
Reserved3	n bytes	N bytes of data. N must be equal to the value of Reserved2.
track number	2 bytes	Index of this song into the list; the first song has position=1
total length	2 bytes	Expected number of songs in the playlist (the track count), or 0
playlist title length	1 byte	Length of the playlist title. Currently reserved; must be set to 0.
playlist title name	n bytes	Optional metadata name of the user-visible title of the playlist. Currently reserved; must be zero-length.

Disc Numbers

A file may also contain a disc number. This identifies which disc it came from when taken from a CD album, for example. Track numbers without an explicit playlist should be assumed to be qualified by the disc number, if present.

Table 1-4 details the fields in a disc number declaration.

Table 1-4 Disc number types

Name	Length	Description
Reserved1	1 byte	Must be set to 0.
Reserved2	1 byte	Length. Must be set to 0.
Reserved3	n bytes	N bytes of data. N must be equal to the value of Reserved2.
Disk Number	2 bytes	The index of this disk into the list; the first disk has position=1
Total Length	2 bytes	The expected number of disks in the album, or 0

Genre

There are two kinds of genres:

- predefined, enumerated
- user-defined

An enumerated genre is stored as a list of 16-bit enumerated values, in which each entry specifies a value from a predefined set of tags. The top byte contains the set identifier, and the lower byte contains the index to a tag in that set. Currently only the ID3 tag set is defined, with a set identifier of 0 (zero).

Note: iTunes tags are one (1) greater than the corresponding ID3 tag. For example, an iTunes genre entry for the genre *classical* (whose ID3 tag value is 32) is specified with the set identifier 0 (indicating ID3) and the tag value 33.

A user-defined genre is stored in textual form (using a suitable textual storage type), as a comma-separated list of names. The user-defined genre format is preferred. Use the enumerated genre format for convenience when modifying a file that already contains an ID3v1 genre tag.

Defined tags

The following tables define the standard types used by iTunes. Where a tag is documented as taking a UTF string, either UTF-8 or UTF-16 may be used, but to keep files compact, UTF-8 should be used unless UTF-16 is needed.

Table 1-5 Defined tags recognized by iTunes

Element	Additional Description	Tag	Data Type
Album Name		©alb	UTF string
Artist		©ART	UTF string
User Comment		©cmt	UTF string
Cover Art	One or more cover art images	covr	JPEG/PNG/BMP data
Copyright		cprt	UTF string
Release Date	YYYY-MM-DD format string (may be incomplete, i.e. only year)	©day	UTF string
Encoded By	Person or company that encoded the recording	©enc	UTF string
Pre-defined Genre	Enumerated value from ID3 tag set, plus 1 (set identifier = 0)	gnre	enum
User Genre	User-specified string	©gen	UTF string
Song Name		©nam	UTF string
Track Sub-Title		©st3	UTF string
Encoding Tool	Software which encoded the recording	©too	UTF string
Composer		©wrt	UTF string
Album Artist	Artist for the whole album (if different than the individual tracks)	aART	UTF string
Disc Compilation	Is disc part of a compilation?	cpil	8-bit "boolean" integer
Disc Number	The data-type 0 is used, implicitly identifying the disk-number record used above	disk	Binary data—see Table 1-4 (page 18)
Grouping	Overall work (like TIT1 in ID3)	grup	UTF string
Content Rating	Does song have explicit content?	rtng	8-bit "boolean" integer
Beats Per Minute		tmpo	32-bit integer
Track Number	The data-type 0 is used, implicitly identifying the track-number record used above	trkn	Binary data—see Table 1-3 (page 17)

The following custom types are under the `com.apple.iTunes` long meaning.

Element	Additional Description	Name	Data Type
Sound Check	Sound Check analysis info	"iTunNORM"	UTF string
Tool Info	Private	"tool"	32-bit integer
CDDB ID1	CDDB TOC (pre-lookup only)	"iTunes_CDDB_1"	UTF string
CDDB ID2	Track Number (pre-lookup only)	"iTunes_CDDB_TrackNumber"	UTF string
CDDB IDs	CDDB MediaID & MuiID	"iTunes_CDDB_IDs"	UTF string

The following tags may also be used; not all software recognizes or interprets them:

Element	Additional Description	Tag	Data Type
Art Director	Person(s) responsible for non-photographic artwork used with content	©ard	UTF string
Arranger	Person(s) responsible for the particular adaptation of composition	©arg	UTF string
Lyricist/Author Name	Writer of the song lyrics	©aut	UTF string
Copyright Acknowledgement(s)	Ackowledgements of those granting permission to use copyrighted material	©cak	UTF string
Conductor	Name of the person who directed the orchestra	©con	UTF string
Song Description	Explanation of the song	©des	UTF string
Director	Name of director for Movie/Video	©dir	UTF string
Equalization preset name	Setting for Equalization of content	©equ	UTF string
Liner Notes	Explanatory notes about a record album, cassette, or compact disk included on the jacket or in the packaging.	©lnt	UTF string
Record Company	Company releasing the song	©mak	UTF string, URL
Original Artist	Name of artist originally attributed with content	©ope	UTF string
Phonogram Rights (P-Line)	Like a copyright, but using the circled P symbol, for audio rights	©phg	UTF string
Producer	Person(s) responsible for creating/supervising the song	©prd	UTF string
Performer	Name/URL of the individual primary members of the band/group	©prf	UTF string, URL
Publisher	Company publishing the song	©pub	UTF string

Element	Additional Description	Tag	Data Type
Sound Engineer	The name of the person doing sound engineering	©sne	UTF string
Soloist	Name of the musician who performs the solo	©sol	UTF string
Credits	Credits for those who provided source content	©src	UTF string
Thanks/Dedications	Notes of acknowledgement/recognition from Artist	©thx	UTF string
Online Extras	Links to content that can only be accessed when connected to the Internet	©url	URL
Executive Producer	Person(s) responsible for creating/supervising the song	©xpd	UTF string

Document Revision History

This table describes the changes to *iTunes Metadata Format Specification*.

Date	Notes
2008-04-16	Preliminary draft.

Document Revision History