welcome: please sign in
location: id3v2.4.0-structure

Informal standard M. Nilsson
Document: id3v2.4.0-structure.txt 1st November 2000

ID3 tag version 2.4.0 - Main Structure

Status of this document

This document is an informal standard and replaces the ID3v2.3.0 standard [ID3v2]. A formal standard will use another revision number even if the content is identical to document. The contents in this document may change for clarifications but never for added or altered functionallity.

Distribution of this document is unlimited.

Abstract

This document describes the main structure of ID3v2.4.0, which is a revised version of the ID3v2 informal standard [ID3v2] version 2.3.0. The ID3v2 offers a flexible way of storing audio meta information within the audio file itself. The information may be technical information, such as equalisation curves, as well as title, performer, copyright etc.

ID3v2.4.0 is meant to be as close as possible to ID3v2.3.0 in order to allow for implementations to be revised as easily as possible.

1. Table of contents

2. Conventions in this document

Text within "" is a text string exactly as it appears in a tag. Numbers preceded with $ are hexadecimal and numbers preceded with % are binary. $xx is used to indicate a byte with unknown content. %x is used to indicate a bit with unknown content. The most significant bit (MSB) of a byte is called 'bit 7' and the least significant bit (LSB) is called 'bit 0'.

A tag is the whole tag described in this document. A frame is a block of information in the tag. The tag consists of a header, frames and optional padding. A field is a piece of information; one value, a string etc. A numeric string is a string that consists of the characters "0123456789" only.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [KEYWORDS].

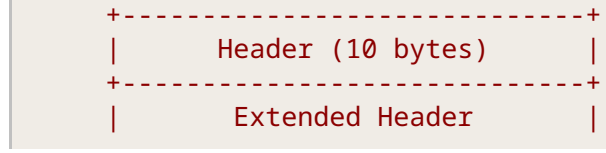
3. ID3v2 overview

ID3v2 is a general tagging format for audio, which makes it possible to store meta data about the audio inside the audio file itself. The ID3 tag described in this document is mainly targeted at files encoded with MPEG-1/2 layer I, MPEG-1/2 layer II, MPEG-1/2 layer III and MPEG-2.5, but may work with other types of encoded audio or as a stand alone format for audio meta data.

ID3v2 is designed to be as flexible and expandable as possible to meet new meta information needs that might arise. To achieve that ID3v2 is constructed as a container for several information blocks, called frames, whose format need not be known to the software that encounters them. At the start of every frame is an unique and predefined identifier, a size descriptor that allows software to skip unknown frames and a flags field. The flags describes encoding details and if the frame should remain in the tag, should it be unknown to the software, if the file is altered.

The bitorder in ID3v2 is most significant bit first (MSB). The byteorder in multibyte numbers is most significant byte first (e.g. $12345678 would be encoded $12 34 56 78), also known as big endian and network byte order.

Overall tag structure:



```
     | (variable length, OPTIONAL) |
     +-----------------------------+
     |   Frames (variable length)  |
     +-----------------------------+
     |           Padding           |
     | (variable length, OPTIONAL) |
     +-----------------------------+
     | Footer (10 bytes, OPTIONAL) |
     +-----------------------------+
```

In general, padding and footer are mutually exclusive. See details in sections 3.3, 3.4 and 5.

3.1. ID3v2 header

The first part of the ID3v2 tag is the 10 byte tag header, laid out as follows:

```
ID3v2/file identifier      "ID3"
ID3v2 version              $04 00
ID3v2 flags                %abcd0000
ID3v2 size             4 * %0xxxxxxx
```

The first three bytes of the tag are always "ID3", to indicate that this is an ID3v2 tag, directly followed by the two version bytes. The first byte of ID3v2 version is its major version, while the second byte is its revision number. In this case this is ID3v2.4.0. All revisions are backwards compatible while major versions are not. If software with ID3v2.4.0 and below support should encounter version five or higher it should simply ignore the whole tag. Version or revision will never be $FF.

The version is followed by the ID3v2 flags field, of which currently four flags are used.

a - Unsynchronisation

Bit 7 in the 'ID3v2 flags' indicates whether or not unsynchronisation is applied on all frames (see section 6.1 for details); a set bit indicates usage.

b - Extended header

The second bit (bit 6) indicates whether or not the header is followed by an extended header. The extended header is described in section 3.2. A set bit indicates the presence of an extended header.

c - Experimental indicator

The third bit (bit 5) is used as an 'experimental indicator'. This flag SHALL always be set when the tag is in an experimental stage.

```
   d - Footer present

     Bit 4 indicates that a footer (section 3.4) is present at the very
     end of the tag. A set bit indicates the presence of a footer.


   All the other flags MUST be cleared. If one of these undefined flags
   are set, the tag might not be readable for a parser that does not
   know the flags function.

   The ID3v2 tag size is stored as a 32 bit synchsafe integer (section
   6.2), making a total of 28 effective bits (representing up to 256MB).

   The ID3v2 tag size is the sum of the byte length of the extended
   header, the padding and the frames after unsynchronisation. If a
   footer is present this equals to ('total size' - 20) bytes, otherwise
   ('total size' - 10) bytes.

   An ID3v2 tag can be detected with the following pattern:
     $49 44 33 yy yy xx zz zz zz zz
   Where yy is less than $FF, xx is the 'flags' byte and zz is less than
   $80.


3.2. Extended header

   The extended header contains information that can provide further
   insight in the structure of the tag, but is not vital to the correct
   parsing of the tag information; hence the extended header is
   optional.

     Extended header size   4 * %0xxxxxxx
     Number of flag bytes       $01
     Extended Flags             $xx

   Where the 'Extended header size' is the size of the whole extended
   header, stored as a 32 bit synchsafe integer. An extended header can
   thus never have a size of fewer than six bytes.

   The extended flags field, with its size described by 'number of flag
   bytes', is defined as:

     %0bcd0000

   Each flag that is set in the extended header has data attached, which
   comes in the order in which the flags are encountered (i.e. the data
   for flag 'b' comes before the data for flag 'c'). Unset flags cannot
   have any attached data. All unknown flags MUST be unset and their
   corresponding data removed when a tag is modified.

   Every set flag's data starts with a length byte, which contains a
   value between 0 and 128 ($00 - $7f), followed by data that has the
   field length indicated by the length byte. If a flag has no attached
   data, the value $00 is used as length byte.
```

```
b - Tag is an update

  If this flag is set, the present tag is an update of a tag found
  earlier in the present file or stream. If frames defined as unique
  are found in the present tag, they are to override any
  corresponding ones found in the earlier tag. This flag has no
  corresponding data.

      Flag data length      $00

c - CRC data present

  If this flag is set, a CRC-32 [ISO-3309] data is included in the
  extended header. The CRC is calculated on all the data between the
  header and footer as indicated by the header's tag length field,
  minus the extended header. Note that this includes the padding (if
  there is any), but excludes the footer. The CRC-32 is stored as an
  35 bit synchsafe integer, leaving the upper four bits always
  zeroed.

     Flag data length       $05
     Total frame CRC    5 * %0xxxxxxx

d - Tag restrictions

  For some applications it might be desired to restrict a tag in more
  ways than imposed by the ID3v2 specification. Note that the
  presence of these restrictions does not affect how the tag is
  decoded, merely how it was restricted before encoding. If this flag
  is set the tag is restricted as follows:

     Flag data length       $01
     Restrictions           %ppqrrstt

  p - Tag size restrictions

    00   No more than 128 frames and 1 MB total tag size.
    01   No more than 64 frames and 128 KB total tag size.
    10   No more than 32 frames and 40 KB total tag size.
    11   No more than 32 frames and 4 KB total tag size.

  q - Text encoding restrictions

    0    No restrictions
    1    Strings are only encoded with ISO-8859-1 [ISO-8859-1] or
         UTF-8 [UTF-8].

  r - Text fields size restrictions

    00   No restrictions
    01   No string is longer than 1024 characters.
    10   No string is longer than 128 characters.
    11   No string is longer than 30 characters.
```

```
      Note that nothing is said about how many bytes is used to
      represent those characters, since it is encoding dependent. If a
      text frame consists of more than one string, the sum of the
      strungs is restricted as stated.

    s - Image encoding restrictions

      0   No restrictions
      1   Images are encoded only with PNG [PNG] or JPEG [JFIF].

    t - Image size restrictions

      00  No restrictions
      01  All images are 256x256 pixels or smaller.
      10  All images are 64x64 pixels or smaller.
      11  All images are exactly 64x64 pixels, unless required
          otherwise.


3.3.   Padding

   It is OPTIONAL to include padding after the final frame (at the end
   of the ID3 tag), making the size of all the frames together smaller
   than the size given in the tag header. A possible purpose of this
   padding is to allow for adding a few additional frames or enlarge
   existing frames within the tag without having to rewrite the entire
   file. The value of the padding bytes must be $00. A tag MUST NOT have
   any padding between the frames or between the tag header and the
   frames. Furthermore it MUST NOT have any padding when a tag footer is
   added to the tag.


3.4.   ID3v2 footer

   To speed up the process of locating an ID3v2 tag when searching from
   the end of a file, a footer can be added to the tag. It is REQUIRED
   to add a footer to an appended tag, i.e. a tag located after all
   audio data. The footer is a copy of the header, but with a different
   identifier.

     ID3v2 identifier           "3DI"
     ID3v2 version              $04 00
     ID3v2 flags                %abcd0000
     ID3v2 size             4 * %0xxxxxxx


4.   ID3v2 frame overview

   All ID3v2 frames consists of one frame header followed by one or more
   fields containing the actual information. The header is always 10
   bytes and laid out as follows:

     Frame ID      $xx xx xx xx  (four characters)
     Size      4 * %0xxxxxxx
     Flags         $xx xx
```

```
The frame ID is made out of the characters capital A-Z and 0-9.
Identifiers beginning with "X", "Y" and "Z" are for experimental
frames and free for everyone to use, without the need to set the
experimental bit in the tag header. Bear in mind that someone else
might have used the same identifier as you. All other identifiers are
either used or reserved for future use.

The frame ID is followed by a size descriptor containing the size of
the data in the final frame, after encryption, compression and
unsynchronisation. The size is excluding the frame header ('total
frame size' - 10 bytes) and stored as a 32 bit synchsafe integer.

In the frame header the size descriptor is followed by two flag
bytes. These flags are described in section 4.1.

There is no fixed order of the frames' appearance in the tag,
although it is desired that the frames are arranged in order of
significance concerning the recognition of the file. An example of
such order: UFID, TIT2, MCDI, TRCK ...

A tag MUST contain at least one frame. A frame must be at least 1
byte big, excluding the header.

If nothing else is said, strings, including numeric strings and URLs
[URL], are represented as ISO-8859-1 [ISO-8859-1] characters in the
range $20 - $FF. Such strings are represented in frame descriptions
as <text string>, or <full text string> if newlines are allowed. If
nothing else is said newline character is forbidden. In ISO-8859-1 a
newline is represented, when allowed, with $0A only.

Frames that allow different types of text encoding contains a text
encoding description byte. Possible encodings:

  $00   ISO-8859-1 [ISO-8859-1]. Terminated with $00.
  $01   UTF-16 [UTF-16] encoded Unicode [UNICODE] with BOM. All
        strings in the same frame SHALL have the same byteorder.
        Terminated with $00 00.
  $02   UTF-16BE [UTF-16] encoded Unicode [UNICODE] without BOM.
        Terminated with $00 00.
  $03   UTF-8 [UTF-8] encoded Unicode [UNICODE]. Terminated with $00.

Strings dependent on encoding are represented in frame descriptions
as <text string according to encoding>, or <full text string
according to encoding> if newlines are allowed. Any empty strings of
type $01 which are NULL-terminated may have the Unicode BOM followed
by a Unicode NULL ($FF FE 00 00 or $FE FF 00 00).

The timestamp fields are based on a subset of ISO 8601. When being as
precise as possible the format of a time string is
yyyy-MM-ddTHH:mm:ss (year, "-", month, "-", day, "T", hour (out of
24), ":", minutes, ":", seconds), but the precision may be reduced by
removing as many time indicators as wanted. Hence valid timestamps
are
yyyy, yyyy-MM, yyyy-MM-dd, yyyy-MM-ddTHH, yyyy-MM-ddTHH:mm and
yyyy-MM-ddTHH:mm:ss. All time stamps are UTC. For durations, use
the slash character as described in 8601, and for multiple non-
```

```
   contiguous dates, use multiple strings, if allowed by the frame
   definition.

   The three byte language field, present in several frames, is used to
   describe the language of the frame's content, according to ISO-639-2
   [ISO-639-2]. The language should be represented in lower case. If the
   language is not known the string "XXX" should be used.

   All URLs [URL] MAY be relative, e.g. "picture.png", "../doc.txt".

   If a frame is longer than it should be, e.g. having more fields than
   specified in this document, that indicates that additions to the
   frame have been made in a later version of the ID3v2 standard. This
   is reflected by the revision number in the header of the tag.


4.1.   Frame header flags

   In the frame header the size descriptor is followed by two flag
   bytes. All unused flags MUST be cleared. The first byte is for
   'status messages' and the second byte is a format description. If an
   unknown flag is set in the first byte the frame MUST NOT be changed
   without that bit cleared. If an unknown flag is set in the second
   byte the frame is likely to not be readable. Some flags in the second
   byte indicates that extra information is added to the header. These
   fields of extra information is ordered as the flags that indicates
   them. The flags field is defined as follows (l and o left out because
   ther resemblence to one and zero):

     %0abc0000 %0h00kmnp

   Some frame format flags indicate that additional information fields
   are added to the frame. This information is added after the frame
   header and before the frame data in the same order as the flags that
   indicates them. I.e. the four bytes of decompressed size will precede
   the encryption method byte. These additions affects the 'frame size'
   field, but are not subject to encryption or compression.

   The default status flags setting for a frame is, unless stated
   otherwise, 'preserved if tag is altered' and 'preserved if file is
   altered', i.e. %00000000.


4.1.1. Frame status flags

   a - Tag alter preservation

     This flag tells the tag parser what to do with this frame if it is
     unknown and the tag is altered in any way. This applies to all
     kinds of alterations, including adding more padding and reordering
     the frames.

     0     Frame should be preserved.
     1     Frame should be discarded.
```

```
   b - File alter preservation

     This flag tells the tag parser what to do with this frame if it is
     unknown and the file, excluding the tag, is altered. This does not
     apply when the audio is completely replaced with other audio data.

     0     Frame should be preserved.
     1     Frame should be discarded.


   c - Read only

      This flag, if set, tells the software that the contents of this
      frame are intended to be read only. Changing the contents might
      break something, e.g. a signature. If the contents are changed,
      without knowledge of why the frame was flagged read only and
      without taking the proper means to compensate, e.g. recalculating
      the signature, the bit MUST be cleared.


4.1.2. Frame format flags

   h - Grouping identity

      This flag indicates whether or not this frame belongs in a group
      with other frames. If set, a group identifier byte is added to the
      frame. Every frame with the same group identifier belongs to the
      same group.

      0     Frame does not contain group information
      1     Frame contains group information


   k - Compression

      This flag indicates whether or not the frame is compressed.
      A 'Data Length Indicator' byte MUST be included in the frame.

      0     Frame is not compressed.
      1     Frame is compressed using zlib [zlib] deflate method.
            If set, this requires the 'Data Length Indicator' bit
            to be set as well.


   m - Encryption

      This flag indicates whether or not the frame is encrypted. If set,
      one byte indicating with which method it was encrypted will be
      added to the frame. See description of the ENCR frame for more
      information about encryption method registration. Encryption
      should be done after compression. Whether or not setting this flag
      requires the presence of a 'Data Length Indicator' depends on the
      specific algorithm used.

      0     Frame is not encrypted.
      1     Frame is encrypted.
```

```
   n - Unsynchronisation

      This flag indicates whether or not unsynchronisation was applied
      to this frame. See section 6 for details on unsynchronisation.
      If this flag is set all data from the end of this header to the
      end of this frame has been unsynchronised. Although desirable, the
      presence of a 'Data Length Indicator' is not made mandatory by
      unsynchronisation.

      0     Frame has not been unsynchronised.
      1     Frame has been unsyrchronised.

   p - Data length indicator

      This flag indicates that a data length indicator has been added to
      the frame. The data length indicator is the value one would write
      as the 'Frame length' if all of the frame format flags were
      zeroed, represented as a 32 bit synchsafe integer.

      0      There is no Data Length Indicator.
      1      A data length Indicator has been added to the frame.


5.   Tag location

   The default location of an ID3v2 tag is prepended to the audio so
   that players can benefit from the information when the data is
   streamed. It is however possible to append the tag, or make a
   prepend/append combination. When deciding upon where an unembedded
   tag should be located, the following order of preference SHOULD be
   considered.

     1. Prepend the tag.

     2. Prepend a tag with all vital information and add a second tag at
        the end of the file, before tags from other tagging systems. The
        first tag is required to have a SEEK frame.

     3. Add a tag at the end of the file, before tags from other tagging
        systems.

   In case 2 and 3 the tag can simply be appended if no other known tags
   are present. The suggested method to find ID3v2 tags are:

     1. Look for a prepended tag using the pattern found in section 3.1.

     2. If a SEEK frame was found, use its values to guide further
        searching.

     3. Look for a tag footer, scanning from the back of the file.

   For every new tag that is found, the old tag should be discarded
   unless the update flag in the extended header (section 3.2) is set.
```

6. Unsynchronisation

The only purpose of unsynchronisation is to make the ID3v2 tag as compatible as possible with existing software and hardware. There is no use in 'unsynchronising' tags if the file is only to be processed only by ID3v2 aware software and hardware. Unsynchronisation is only useful with tags in MPEG 1/2 layer I, II and III, MPEG 2.5 and AAC files.

6.1. The unsynchronisation scheme

Whenever a false synchronisation is found within the tag, one zeroed byte is inserted after the first false synchronisation byte. The format of synchronisations that should be altered by ID3 encoders is as follows:

```
%11111111 111xxxxx
```

and should be replaced with:

```
%11111111 00000000 111xxxxx
```

This has the side effect that all $FF 00 combinations have to be altered, so they will not be affected by the decoding process. Therefore all the $FF 00 combinations have to be replaced with the $FF 00 00 combination during the unsynchronisation.

To indicate usage of the unsynchronisation, the unsynchronisation flag in the frame header should be set. This bit MUST be set if the frame was altered by the unsynchronisation and SHOULD NOT be set if unaltered. If all frames in the tag are unsynchronised the unsynchronisation flag in the tag header SHOULD be set. It MUST NOT be set if the tag has a frame which is not unsynchronised.

Assume the first byte of the audio to be $FF. The special case when the last byte of the last frame is $FF and no padding nor footer is used will then introduce a false synchronisation. This can be solved by adding a footer, adding padding or unsynchronising the frame and add $00 to the end of the frame data, thus adding more byte to the frame size than a normal unsynchronisation would. Although not preferred, it is allowed to apply the last method on all frames ending with $FF.

It is preferred that the tag is either completely unsynchronised or not unsynchronised at all. A completely unsynchronised tag has no false synchonisations in it, as defined above, and does not end with $FF. A completely non-unsynchronised tag contains no unsynchronised frames, and thus the unsynchronisation flag in the header is cleared.

Do bear in mind, that if compression or encryption is used, the unsynchronisation scheme MUST be applied afterwards. When decoding an unsynchronised frame, the unsynchronisation scheme MUST be reversed first, encryption and decompression afterwards.

```
6.2.   Synchsafe integers

   In some parts of the tag it is inconvenient to use the
   unsychronisation scheme because the size of unsynchronised data is
   not known in advance, which is particularly problematic with size
   descriptors. The solution in ID3v2 is to use synchsafe integers, in
   which there can never be any false synchs. Synchsafe integers are
   integers that keep its highest bit (bit 7) zeroed, making seven bits
   out of eight available. Thus a 32 bit synchsafe integer can store 28
   bits of information.

   Example:

     255 (%11111111) encoded as a 16 bit synchsafe integer is 383
     (%00000001 01111111).


7.   Copyright

   Copyright (C) Martin Nilsson 2000. All Rights Reserved.

   This document and translations of it may be copied and furnished to
   others, and derivative works that comment on or otherwise explain it
   or assist in its implementation may be prepared, copied, published
   and distributed, in whole or in part, without restriction of any
   kind, provided that a reference to this document is included on all
   such copies and derivative works. However, this document itself may
   not be modified in any way and reissued as the original document.

   The limited permissions granted above are perpetual and will not be
   revoked.

   This document and the information contained herein is provided on an
   'AS IS' basis and THE AUTHORS DISCLAIMS ALL WARRANTIES, EXPRESS OR
   IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF
   THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED
   WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.


8.   References

   [ID3v2] Martin Nilsson, 'ID3v2 informal standard'.

      <url:http://www.id3.org/id3v2.3.0.txt>

   [ISO-639-2] ISO/FDIS 639-2.
   'Codes for the representation of names of languages, Part 2: Alpha-3
   code.' Technical committee / subcommittee: TC 37 / SC 2

   [ISO-3309] ISO 3309
   'Information Processing Systems--Data Communication High-Level Data
   Link Control Procedure--Frame Structure', IS 3309, October 1984, 3rd
   Edition.

   [ISO-8859-1] ISO/IEC DIS 8859-1.
   '8-bit single-byte coded graphic character sets, Part 1: Latin
```

```
alphabet No. 1.' Technical committee / subcommittee: JTC 1 / SC 2

[JFIF] 'JPEG File Interchange Format, version 1.02'

   <url:http://www.w3.org/Graphics/JPEG/jfif.txt>

[KEYWORDS] S. Bradner, 'Key words for use in RFCs to Indicate
Requirement Levels', RFC 2119, March 1997.

   <url:ftp://ftp.isi.edu/in-notes/rfc2119.txt>

[MPEG] ISO/IEC 11172-3:1993.
'Coding of moving pictures and associated audio for digital storage
media at up to about 1,5 Mbit/s, Part 3: Audio.'
Technical committee / subcommittee: JTC 1 / SC 29
 and
ISO/IEC 13818-3:1995
'Generic coding of moving pictures and associated audio information,
Part 3: Audio.'
Technical committee / subcommittee: JTC 1 / SC 29
 and
ISO/IEC DIS 13818-3
'Generic coding of moving pictures and associated audio information,
Part 3: Audio (Revision of ISO/IEC 13818-3:1995)'

[PNG] 'Portable Network Graphics, version 1.0'

   <url:http://www.w3.org/TR/REC-png-multi.html>

[UNICODE] The Unicode Consortium,
'The Unicode Standard Version 3.0', ISBN 0-201-61633-5.

<url:http://www.unicode.org/unicode/standard/versions/Unicode3.0.htm>

[URL] T. Berners-Lee, L. Masinter & M. McCahill, 'Uniform Resource
Locators (URL)', RFC 1738, December 1994.

   <url:ftp://ftp.isi.edu/in-notes/rfc1738.txt>

[UTF-8] F. Yergeau, 'UTF-8, a transformation format of ISO 10646',
RFC 2279, January 1998.

   <url:ftp://ftp.isi.edu/in-notes/rfc2279.txt>

[UTF-16] F. Yergeau, 'UTF-16, an encoding of ISO 10646', RFC 2781,
February 2000.

   <url:ftp://ftp.isi.edu/in-notes/rfc2781.txt>

[ZLIB] P. Deutsch, Aladdin Enterprises & J-L. Gailly, 'ZLIB
Compressed Data Format Specification version 3.3', RFC 1950,
May 1996.

   <url:ftp://ftp.isi.edu/in-notes/rfc1950.txt>
```

```
9.   Author's Address

   Written by

     Martin Nilsson
     Rydsv�gen 246 C. 30
     SE-584 34 Link�ping
     Sweden

     Email: nilsson at id3.org
```

id3v2.4.0-structure (last edited 2012-10-08 22:15:41 by localhost)