Mutagen Specs

Release 1.0

January 06, 2017

Contents

CHAPTER 1

ID3

1.1 ID3 tag version 2

1.1.1 Status of this document

This document is an Informal standard and is released so that implementors could have a set standard before the formal standard is set. The formal standard will use another version number if not identical to what is described in this document. The contents in this document may change for clarifications but never for added or altered functionallity.

Distribution of this document is unlimited.

1.1.2 Abstract

The recent gain of popularity for MPEG layer III audio files on the internet forced a standardised way of storing information about an audio file within itself to determinate its origin and contents.

Today the most accepted way to do this is with the so called ID3 tag, which is simple but very limited and in some cases very unsuitable. The ID3 tag has very limited space in every field, very limited numbers of fields, not expandable or upgradeable and is placed at the end of a the file, which is unsuitable for streaming audio. This draft is an attempt to answer these issues with a new version of the ID3 tag.

1.1.3 Conventions in this document

In the examples, text within “” is a text string exactly as it appears in a file. 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 0-9 only.

1.1.4 ID3v2 overview

The two biggest design goals were to be able to implement ID3v2 without disturbing old software too much and that ID3v2 should be expandable.

The first criterion is met by the simple fact that the MPEG [MPEG] decoding software uses a syncsignal, embedded in the audiostream, to ‘lock on to’ the audio. Since the ID3v2 tag doesn’t contain a valid syncsignal, no software will

attempt to play the tag. If, for any reason, coincidence make a syncsignal appear within the tag it will be taken care of by the 'unsynchronisation scheme' described in section 5.

The second criterion has made a more noticeable impact on the design of the ID3v2 tag. It 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 there is an identifier that explains the frames's format and content, and a size descriptor that allows software to skip unknown frames.

If a total revision of the ID3v2 tag should be needed, there is a version number and a size descriptor in the ID3v2 header.

The ID3 tag described in this document is mainly targeted to files encoded with MPEG-2 layer I, MPEG-2 layer II, MPEG-2 layer III and MPEG-2.5, but may work with other types of encoded audio.

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).

It is permitted to include padding after all 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 head of the tag. 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.

1.1.5 ID3v2 header

The ID3v2 tag header, which should be the first information in the file, is 10 bytes as follows:

```
ID3/file identifier      "ID3"
ID3 version              $02 00
ID3 flags                %xx000000
ID3 size             4 * %0xxxxxxx
```

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

The first bit (bit 7) in the 'ID3 flags' is indicating whether or not unsynchronisation is used (see section 5 for details); a set bit indicates usage.

The second bit (bit 6) is indicating whether or not compression is used; a set bit indicates usage. Since no compression scheme has been decided yet, the ID3 decoder (for now) should just ignore the entire tag if the compression bit is set.

The ID3 tag size is encoded with four bytes where the first bit (bit 7) is set to zero in every byte, making a total of 28 bits. The zeroed bits are ignored, so a 257 bytes long tag is represented as $00 00 02 01.

The ID3 tag size is the size of the complete tag after unsychronisation, including padding, excluding the header (total tag size - 10). The reason to use 28 bits (representing up to 256MB) for size description is that we don't want to run out of space here.

A 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.

1.1.6 ID3v2 frames overview

The headers of the frames are similar in their construction. They consist of one three character identifier (capital A-Z and 0-9) and one three byte size field, making a total of six bytes. The header is excluded from the size. Identifiers

beginning with “X”, “Y” and “Z” are for experimental use and free for everyone to use. Have 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 three character frame identifier is followed by a three byte size descriptor, making a total header size of six bytes in every frame. The size is calculated as framesize excluding frame identifier and size descriptor (frame size - 6).

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: UFI, MCI, TT2 ...

A tag must contain at least one frame. A frame must be at least 1 byte big, excluding the 6-byte header.

If nothing else is said a string is represented as ISO-8859-1 [ISO-8859-1] characters in the range $20 - $FF. All unicode strings [UNICODE] use 16-bit unicode 2.0 (ISO/IEC 10646-1:1993, UCS-2). All numeric strings are always encoded as ISO-8859-1. Terminated strings are terminated with $00 if encoded with ISO-8859-1 and $00 00 if encoded as unicode. If nothing else is said newline character is forbidden. In ISO-8859-1 a new line is represented, when allowed, with $0A only. Frames that allow different types of text encoding have a text encoding description byte directly after the frame size. If ISO-8859-1 is used this byte should be $00, if unicode is used it should be $01.

The three byte language field is used to describe the language of the frame’s content, according to ISO-639-2 [ISO-639-2].

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 ID3 standard. This is reflected by the revision number in the header of the tag.

1.1.7 Declared ID3v2 frames

The following frames are declared in this draft.

- BUF Recommended buffer size
- CNT Play counter
- COM Comments
- CRA Audio encryption
- CRM Encrypted meta frame
- ETC Event timing codes
- EQU Equalization
- GEO General encapsulated object
- IPL Involved people list
- LNK Linked information
- MCI Music CD Identifier
- MLL MPEG location lookup table
- PIC Attached picture
- POP Popularimeter
- REV Reverb
- RVA Relative volume adjustment
- SLT Synchronized lyric/text
- STC Synced tempo codes

- TAL Album/Movie/Show title
- TBP BPM (Beats Per Minute)
- TCM Composer
- TCO Content type
- TCR Copyright message
- TDA Date
- TDY Playlist delay
- TEN Encoded by
- TFT File type
- TIM Time
- TKE Initial key
- TLA Language(s)
- TLE Length
- TMT Media type
- TOA Original artist(s)/performer(s)
- TOF Original filename
- TOL Original Lyricist(s)/text writer(s)
- TOR Original release year
- TOT Original album/Movie/Show title
- TP1 Lead artist(s)/Lead performer(s)/Soloist(s)/Performing group
- TP2 Band/Orchestra/Accompaniment
- TP3 Conductor/Performer refinement
- TP4 Interpreted, remixed, or otherwise modified by
- TPA Part of a set
- TPB Publisher
- TRC ISRC (International Standard Recording Code)
- TRD Recording dates
- TRK Track number/Position in set
- TSI Size
- TSS Software/hardware and settings used for encoding
- TT1 Content group description
- TT2 Title/Songname/Content description
- TT3 Subtitle/Description refinement
- TXT Lyricist/text writer
- TXX User defined text information frame
- TYE Year

- UFI Unique file identifier
- ULT Unsychronized lyric/text transcription
- WAF Official audio file webpage
- WAR Official artist/performer webpage
- WAS Official audio source webpage
- WCM Commercial information
- WCP Copyright/Legal information
- WPB Publishers official webpage
- WXX User defined URL link frame

1.1.8 Unique file identifier

This frame's purpose is to be able to identify the audio file in a database that may contain more information relevant to the content. Since standardisation of such a database is beyond this document, all frames begin with a null-terminated string with a URL [URL] containing an email address, or a link to a location where an email address can be found, that belongs to the organisation responsible for this specific database implementation. Questions regarding the database should be sent to the indicated email address. The URL should not be used for the actual database queries. If a $00 is found directly after the 'Frame size' the whole frame should be ignored, and preferably be removed. The 'Owner identifier' is then followed by the actual identifier, which may be up to 64 bytes. There may be more than one "UFI" frame in a tag, but only one with the same 'Owner identifier'.

```
Unique file identifier  "UFI"
Frame size              $xx xx xx
Owner identifier        <textstring> $00
Identifier              <up to 64 bytes binary data>
```

1.1.9 Text information frames

The text information frames are the most important frames, containing information like artist, album and more. There may only be one text information frame of its kind in an tag. If the textstring is followed by a termination ($00 (00)) all the following information should be ignored and not be displayed. All the text information frames have the following format:

```
Text information identifier  "T00" - "TZZ" , excluding "TXX",
                           described in 4.2.2.
Frame size                   $xx xx xx
Text encoding                $xx
Information                  <textstring>
```

1.1.10 Text information frames - details

TT1 The 'Content group description' frame is used if the sound belongs to a larger category of sounds/music. For example, classical music is often sorted in different musical sections (e.g. "Piano Concerto", "Weather - Hurricane").

TT2 The 'Title/Songname/Content description' frame is the actual name of the piece (e.g. "Adagio", "Hurricane Donna").

TT3 The 'Subtitle/Description refinement' frame is used for information directly related to the contents title (e.g. "Op. 16" or "Performed live at wembley").

TP1 The 'Lead artist(s)/Lead performer(s)/Soloist(s)/Performing group' is used for the main artist(s). They are seperated with the "/" character.

TP2 The 'Band/Orchestra/Accompaniment' frame is used for additional information about the performers in the recording.

TP3 The 'Conductor' frame is used for the name of the conductor.

TP4 The 'Interpreted, remixed, or otherwise modified by' frame contains more information about the people behind a remix and similar interpretations of another existing piece.

TCM The 'Composer(s)' frame is intended for the name of the composer(s). They are seperated with the "/" character.

TXT The 'Lyricist(s)/text writer(s)' frame is intended for the writer(s) of the text or lyrics in the recording. They are seperated with the "/" character.

TLA The 'Language(s)' frame should contain the languages of the text or lyrics in the audio file. The language is represented with three characters according to ISO-639-2. If more than one language is used in the text their language codes should follow according to their usage.

TCO The content type, which previously (in ID3v1.1, see appendix A) was stored as a one byte numeric value only, is now a numeric string. You may use one or several of the types as ID3v1.1 did or, since the category list would be impossible to maintain with accurate and up to date categories, define your own. References to the ID3v1 genres can be made by, as first byte, enter "(" followed by a number from the genres list (section A.3.) and ended with a ")" character. This is optionally followed by a refinement, e.g. "(21)" or "(4)Eurodisco". Several references can be made in the same frame, e.g. "(51)(39)". If the refinement should begin with a "(" character it should be replaced with "((", e.g. "((I can figure out any genre)" or "(55)((I think...)". The following new content types is defined in ID3v2 and is implemented in the same way as the numerig content types, e.g. "(RX)".

    RX Remix CR Cover

TAL The 'Album/Movie/Show title' frame is intended for the title of the recording(/source of sound) which the audio in the file is taken from.

TPA The 'Part of a set' frame is a numeric string that describes which part of a set the audio came from. This frame is used if the source described in the "TAL" frame is divided into several mediums, e.g. a double CD. The value may be extended with a "/" character and a numeric string containing the total number of parts in the set. E.g. "1/2".

TRK The 'Track number/Position in set' frame is a numeric string containing the order number of the audio-file on its original recording. This may be extended with a "/" character and a numeric string containing the total numer of tracks/elements on the original recording. E.g. "4/9".

TRC The 'ISRC' frame should contian the International Standard Recording Code [ISRC].

TYE The 'Year' frame is a numeric string with a year of the recording. This frames is always four characters long (until the year 10000).

TDA The 'Date' frame is a numeric string in the DDMM format containing the date for the recording. This field is always four characters long.

TIM The 'Time' frame is a numeric string in the HHMM format containing the time for the recording. This field is always four characters long.

TRD The 'Recording dates' frame is a intended to be used as complement to the "TYE", "TDA" and "TIM" frames. E.g. "4th-7th June, 12th June" in combination with the "TYE" frame.

TMT The 'Media type' frame describes from which media the sound originated. This may be a textstring or a reference to the predefined media types found in the list below. References are made within "(" and ")" and are optionally followed by a text refinement, e.g. "(MC) with four channels". If a text refinement should begin with

a "(" character it should be replaced with "((" in the same way as in the "TCO" frame. Predefined refinements is appended after the media type, e.g. "(CD/S)" or "(VID/PAL/VHS)".

```
DIG    Other digital media
  /A    Analog transfer from media

ANA    Other analog media
  /WAC  Wax cylinder
  /8CA  8-track tape cassette

CD     CD
  /A    Analog transfer from media
  /DD   DDD
  /AD   ADD
  /AA   AAD

LD     Laserdisc
  /A     Analog transfer from media

TT     Turntable records
  /33    33.33 rpm
  /45    45 rpm
  /71    71.29 rpm
  /76    76.59 rpm
  /78    78.26 rpm
  /80    80 rpm

MD     MiniDisc
  /A    Analog transfer from media

DAT    DAT
  /A    Analog transfer from media
  /1    standard, 48 kHz/16 bits, linear
  /2    mode 2, 32 kHz/16 bits, linear
  /3    mode 3, 32 kHz/12 bits, nonlinear, low speed
  /4    mode 4, 32 kHz/12 bits, 4 channels
  /5    mode 5, 44.1 kHz/16 bits, linear
  /6    mode 6, 44.1 kHz/16 bits, 'wide track' play

DCC    DCC
  /A    Analog transfer from media

DVD    DVD
  /A    Analog transfer from media

TV     Television
  /PAL    PAL
  /NTSC   NTSC
  /SECAM  SECAM

VID    Video
  /PAL    PAL
  /NTSC   NTSC
  /SECAM  SECAM
  /VHS    VHS
  /SVHS   S-VHS
  /BETA   BETAMAX

RAD    Radio
```

```
    /FM   FM
    /AM   AM
    /LW   LW
    /MW   MW

  TEL    Telephone
    /I    ISDN

  MC     MC (normal cassette)
    /4    4.75 cm/s (normal speed for a two sided cassette)
    /9    9.5 cm/s
    /I    Type I cassette (ferric/normal)
    /II   Type II cassette (chrome)
    /III  Type III cassette (ferric chrome)
    /IV   Type IV cassette (metal)

  REE    Reel
    /9    9.5 cm/s
    /19   19 cm/s
    /38   38 cm/s
    /76   76 cm/s
    /I    Type I cassette (ferric/normal)
    /II   Type II cassette (chrome)
    /III  Type III cassette (ferric chrome)
    /IV   Type IV cassette (metal)
```

TFT

The 'File type' frame indicates which type of audio this tag defines. The following type and refinements are defined:

```
      MPG    MPEG Audio
        /1     MPEG 2 layer I
        /2     MPEG 2 layer II
        /3     MPEG 2 layer III
        /2.5   MPEG 2.5
        /AAC   Advanced audio compression
```

but other types may be used, not for these types though. This is used in a similar way to the predefined types in the "TMT" frame, but without parenthesis. If this frame is not present audio type is assumed to be "MPG".

TBP BPM is short for beats per minute, and is easily computed by dividing the number of beats in a musical piece with its length. To get a more accurate result, do the BPM calculation on the main-part only. To acquire best result measure the time between each beat and calculate individual BPM for each beat and use the median value as result. BPM is an integer and represented as a numerical string.

TCR The 'Copyright message' frame, which must begin with a year and a space character (making five characters), is intended for the copyright holder of the original sound, not the audio file itself. The absence of this frame means only that the copyright information is unavailable or has been removed, and must not be interpreted to mean that the sound is public domain. Every time this field is displayed the field must be preceded with "Copyright " (C) " ", where (C) is one character showing a C in a circle.

TPB The 'Publisher' frame simply contains the name of the label or publisher.

TEN The 'Encoded by' frame contains the name of the person or organisation that encoded the audio file. This field may contain a copyright message, if the audio file also is copyrighted by the encoder.

TSS The 'Software/hardware and settings used for encoding' frame includes the used audio encoder and its settings when the file was encoded. Hardware refers to hardware encoders, not the computer on which a program was run.

TOF The 'Original filename' frame contains the preferred filename for the file, since some media doesn't allow the desired length of the filename. The filename is case sensitive and includes its suffix.

TLE The 'Length' frame contains the length of the audiofile in milliseconds, represented as a numeric string.

TSI The 'Size' frame contains the size of the audiofile in bytes excluding the tag, represented as a numeric string.

TDY The 'Playlist delay' defines the numbers of milliseconds of silence between every song in a playlist. The player should use the "ETC" frame, if present, to skip initial silence and silence at the end of the audio to match the 'Playlist delay' time. The time is represented as a numeric string.

TKE The 'Initial key' frame contains the musical key in which the sound starts. It is represented as a string with a maximum length of three characters. The ground keys are represented with "A","B","C","D","E", "F" and "G" and halfkeys represented with "b" and "#". Minor is represented as "m". Example "Cbm". Off key is represented with an "o" only.

TOT The 'Original album/Movie/Show title' frame is intended for the title of the original recording(/source of sound), if for example the music in the file should be a cover of a previously released song.

TOA The 'Original artist(s)/performer(s)' frame is intended for the performer(s) of the original recording, if for example the music in the file should be a cover of a previously released song. The performers are seperated with the "/" character.

TOL The 'Original Lyricist(s)/text writer(s)' frame is intended for the text writer(s) of the original recording, if for example the music in the file should be a cover of a previously released song. The text writers are seperated with the "/" character.

TOR The 'Original release year' frame is intended for the year when the original recording, if for example the music in the file should be a cover of a previously released song, was released. The field is formatted as in the "TDY" frame.

1.1.11 User defined text information frame

This frame is intended for one-string text information concerning the audiofile in a similar way to the other "T"xx frames. The frame body consists of a description of the string, represented as a terminated string, followed by the actual string. There may be more than one "TXX" frame in each tag, but only one with the same description.

```
User defined...   "TXX"
Frame size        $xx xx xx
Text encoding     $xx
Description       <textstring> $00 (00)
Value             <textstring>
```

1.1.12 URL link frames

With these frames dynamic data such as webpages with touring information, price information or plain ordinary news can be added to the tag. There may only be one URL [URL] link frame of its kind in an tag, except when stated otherwise in the frame description. If the textstring is followed by a termination ($00 (00)) all the following information should be ignored and not be displayed. All URL link frames have the following format:

```
URL link frame    "W00" - "WZZ" , excluding "WXX"
                                (described in 4.3.2.)
Frame size        $xx xx xx
URL               <textstring>
```

1.1.13 URL link frames - details

WAF The 'Official audio file webpage' frame is a URL pointing at a file specific webpage.

WAR The 'Official artist/performer webpage' frame is a URL pointing at the artists official webpage. There may be more than one "WAR" frame in a tag if the audio contains more than one performer.

WAS The 'Official audio source webpage' frame is a URL pointing at the official webpage for the source of the audio file, e.g. a movie.

WCM The 'Commercial information' frame is a URL pointing at a webpage with information such as where the album can be bought. There may be more than one "WCM" frame in a tag.

WCP The 'Copyright/Legal information' frame is a URL pointing at a webpage where the terms of use and ownership of the file is described.

WPB The 'Publishers official webpage' frame is a URL pointing at the official wepage for the publisher.

1.1.14 User defined URL link frame

This frame is intended for URL [URL] links concerning the audiofile in a similar way to the other "W"xx frames. The frame body consists of a description of the string, represented as a terminated string, followed by the actual URL. The URL is always encoded with ISO-8859-1 [ISO-8859-1]. There may be more than one "WXX" frame in each tag, but only one with the same description.

```
User defined...   "WXX"
Frame size        $xx xx xx
Text encoding     $xx
Description       <textstring> $00 (00)
URL               <textstring>
```

1.1.15 Involved people list

Since there might be a lot of people contributing to an audio file in various ways, such as musicians and technicians, the 'Text information frames' are often insufficient to list everyone involved in a project. The 'Involved people list' is a frame containing the names of those involved, and how they were involved. The body simply contains a terminated string with the involvement directly followed by a terminated string with the involvee followed by a new involvement and so on. There may only be one "IPL" frame in each tag.

```
Involved people list   "IPL"
Frame size             $xx xx xx
Text encoding          $xx
People list strings    <textstrings>
```

1.1.16 Music CD Identifier

This frame is intended for music that comes from a CD, so that the CD can be identified in databases such as the CDDB [CDDB]. The frame consists of a binary dump of the Table Of Contents, TOC, from the CD, which is a header of 4 bytes and then 8 bytes/track on the CD making a maximum of 804 bytes. This frame requires a present and valid "TRK" frame. There may only be one "MCI" frame in each tag.

```
Music CD identifier   "MCI"
Frame size            $xx xx xx
CD TOC                <binary data>
```

1.1.17 Event timing codes

This frame allows synchronisation with key events in a song or sound. The head is:

```
Event timing codes   "ETC"
Frame size           $xx xx xx
Time stamp format    $xx
```

Where time stamp format is:

```
$01  Absolute time, 32 bit sized, using MPEG [MPEG] frames as unit
$02  Absolute time, 32 bit sized, using milliseconds as unit
```

Abolute time means that every stamp contains the time from the beginning of the file.

Followed by a list of key events in the following format:

```
Type of event   $xx
Time stamp      $xx (xx ...)
```

The 'Time stamp' is set to zero if directly at the beginning of the sound or after the previous event. All events should be sorted in chronological order. The type of event is as follows:

```
$00  padding (has no meaning)
$01  end of initial silence
$02  intro start
$03  mainpart start
$04  outro start
$05  outro end
$06  verse begins
$07  refrain begins
$08  interlude
$09  theme start
$0A  variation
$0B  key change
$0C  time change
$0D  unwanted noise (Snap, Crackle & Pop)

$0E-$DF  reserved for future use

$E0-$EF  not predefined sync 0-F

$F0-$FC  reserved for future use

$FD  audio end (start of silence)
$FE  audio file ends
$FF  one more byte of events follows (all the following bytes with
     the value $FF have the same function)
```

The 'Not predefined sync's ($E0-EF) are for user events. You might want to synchronise your music to something, like setting of an explosion on-stage, turning on your screensaver etc.

There may only be one "ETC" frame in each tag.

1.1.18 MPEG location lookup table

To increase performance and accuracy of jumps within a MPEG [MPEG] audio file, frames with timecodes in different locations in the file might be useful. The ID3 frame includes references that the software can use to calculate positions in the file. After the frame header is a descriptor of how much the 'frame counter' should increase for every reference.

If this value is two then the first reference points out the second frame, the 2nd reference the 4th frame, the 3rd reference the 6th frame etc. In a similar way the 'bytes between reference' and 'milliseconds between reference' points out bytes and milliseconds respectively.

Each reference consists of two parts; a certain number of bits, as defined in 'bits for bytes deviation', that describes the difference between what is said in 'bytes between reference' and the reality and a certain number of bits, as defined in 'bits for milliseconds deviation', that describes the difference between what is said in 'milliseconds between reference' and the reality. The number of bits in every reference, i.e. 'bits for bytes deviation'+'bits for milliseconds deviation', must be a multiple of four. There may only be one "MLL" frame in each tag.

```
Location lookup table          "MLL"
ID3 frame size                 $xx xx xx
MPEG frames between reference  $xx xx
Bytes between reference        $xx xx xx
Milliseconds between reference $xx xx xx
Bits for bytes deviation       $xx
Bits for milliseconds dev.     $xx
```

Then for every reference the following data is included;

```
Deviation in bytes         %xxx....
Deviation in milliseconds  %xxx....
```

1.1.19 Synced tempo codes

For a more accurate description of the tempo of a musical piece this frame might be used. After the header follows one byte describing which time stamp format should be used. Then follows one or more tempo codes. Each tempo code consists of one tempo part and one time part. The tempo is in BPM described with one or two bytes. If the first byte has the value $FF, one more byte follows, which is added to the first giving a range from 2 - 510 BPM, since $00 and $01 is reserved. $00 is used to describe a beat-free time period, which is not the same as a music-free time period. $01 is used to indicate one single beat-stroke followed by a beat-free period.

The tempo descriptor is followed by a time stamp. Every time the tempo in the music changes, a tempo descriptor may indicate this for the player. All tempo descriptors should be sorted in chronological order. The first beat-stroke in a time-period is at the same time as the beat description occurs. There may only be one "STC" frame in each tag.

```
Synced tempo codes  "STC"
Frame size          $xx xx xx
Time stamp format   $xx
Tempo data          <binary data>
```

Where time stamp format is:

```
$01  Absolute time, 32 bit sized, using MPEG [MPEG] frames as unit
$02  Absolute time, 32 bit sized, using milliseconds as unit
```

Abolute time means that every stamp contains the time from the beginning of the file.

1.1.20 Unsychronised lyrics/text transcription

This frame contains the lyrics of the song or a text transcription of other vocal activities. The head includes an encoding descriptor and a content descriptor. The body consists of the actual text. The 'Content descriptor' is a terminated string. If no descriptor is entered, 'Content descriptor' is $00 (00) only. Newline characters are allowed in the text. Maximum length for the descriptor is 64 bytes. There may be more than one lyrics/text frame in each tag, but only one with the same language and content descriptor.

```
Unsynced lyrics/text "ULT"
Frame size           $xx xx xx
Text encoding        $xx
Language             $xx xx xx
Content descriptor   <textstring> $00 (00)
Lyrics/text          <textstring>
```

1.1.21 Synchronised lyrics/text

This is another way of incorporating the words, said or sung lyrics, in the audio file as text, this time, however, in sync with the audio. It might also be used to describing events e.g. occurring on a stage or on the screen in sync with the audio. The header includes a content descriptor, represented with as terminated textstring. If no descriptor is entered, 'Content descriptor' is $00 (00) only.

```
Synced lyrics/text   "SLT"
Frame size           $xx xx xx
Text encoding        $xx
Language             $xx xx xx
Time stamp format    $xx
Content type         $xx
Content descriptor   <textstring> $00 (00)
```

Encoding: $00 ISO-8859-1 [ISO-8859-1] character set is used => $00

is sync identifier.

$01 Unicode [UNICODE] character set is used => $00 00 is sync identifier.

Content type: $00 is other $01 is lyrics $02 is text transcription $03 is movement/part name (e.g. "Adagio") $04 is events (e.g. "Don Quijote enters the stage") $05 is chord (e.g. "Bb F Fsus")

Time stamp format is:

```
$01  Absolute time, 32 bit sized, using MPEG [MPEG] frames as unit
$02  Absolute time, 32 bit sized, using milliseconds as unit
```

Abolute time means that every stamp contains the time from the beginning of the file.

The text that follows the frame header differs from that of the unsynchronised lyrics/text transcription in one major way. Each syllable (or whatever size of text is considered to be convenient by the encoder) is a null terminated string followed by a time stamp denoting where in the sound file it belongs. Each sync thus has the following structure:

```
Terminated text to be synced (typically a syllable)
Sync identifier (terminator to above string)   $00 (00)
Time stamp                                     $xx (xx ...)
```

The 'time stamp' is set to zero or the whole sync is omitted if located directly at the beginning of the sound. All time stamps should be sorted in chronological order. The sync can be considered as a validator of the subsequent string.

Newline characters are allowed in all "SLT" frames and should be used after every entry (name, event etc.) in a frame with the content type $03 - $04.

A few considerations regarding whitespace characters: Whitespace separating words should mark the beginning of a new word, thus occurring in front of the first syllable of a new word. This is also valid for new line characters. A syllable followed by a comma should not be broken apart with a sync (both the syllable and the comma should be before the sync).

An example: The "ULT" passage

```
"Strangers in the night" $0A "Exchanging glances"
```

would be "SLT" encoded as:

```
"Strang" $00 xx xx "ers" $00 xx xx " in" $00 xx xx " the" $00 xx xx
" night" $00 xx xx 0A "Ex" $00 xx xx "chang" $00 xx xx "ing" $00 xx
xx "glan" $00 xx xx "ces" $00 xx xx
```

There may be more than one "SLT" frame in each tag, but only one with the same language and content descriptor.

1.1.22 Comments

This frame replaces the old 30-character comment field in ID3v1. It consists of a frame head followed by encoding, language and content descriptors and is ended with the actual comment as a text string. Newline characters are allowed in the comment text string. There may be more than one comment frame in each tag, but only one with the same language and content descriptor.

```
Comment                    "COM"
Frame size                 $xx xx xx
Text encoding              $xx
Language                   $xx xx xx
Short content description <textstring> $00 (00)
The actual text            <textstring>
```

1.1.23 Relative volume adjustment

This is a more subjective function than the previous ones. It allows the user to say how much he wants to increase/decrease the volume on each channel while the file is played. The purpose is to be able to align all files to a reference volume, so that you don't have to change the volume constantly. This frame may also be used to balance adjust the audio. If the volume peak levels are known then this could be described with the 'Peak volume right' and 'Peak volume left' field. If Peakvolume is not known these fields could be left zeroed or completely omitted. There may only be one "RVA" frame in each tag.

```
Relative volume adjustment    "RVA"
Frame size                    $xx xx xx
Increment/decrement           %000000xx
Bits used for volume descr.   $xx
Relative volume change, right $xx xx (xx ...)
Relative volume change, left  $xx xx (xx ...)
Peak volume right             $xx xx (xx ...)
Peak volume left              $xx xx (xx ...)
```

In the increment/decrement field bit 0 is used to indicate the right channel and bit 1 is used to indicate the left channel. 1 is increment and 0 is decrement.

The 'bits used for volume description' field is normally $10 (16 bits) for MPEG 2 layer I, II and III [MPEG] and MPEG 2.5. This value may not be $00. The volume is always represented with whole bytes, padded in the beginning (highest bits) when 'bits used for volume description' is not a multiple of eight.

1.1.24 Equalisation

This is another subjective, alignment frame. It allows the user to predefine an equalisation curve within the audio file. There may only be one "EQU" frame in each tag.

```
Equalisation       "EQU"
Frame size         $xx xx xx
Adjustment bits    $xx
```

The 'adjustment bits' field defines the number of bits used for representation of the adjustment. This is normally $10 (16 bits) for MPEG 2 layer I, II and III [MPEG] and MPEG 2.5. This value may not be $00.

This is followed by 2 bytes + ('adjustment bits' rounded up to the nearest byte) for every equalisation band in the following format, giving a frequency range of 0 - 32767Hz:

```
Increment/decrement   %x (MSB of the Frequency)
Frequency             (lower 15 bits)
Adjustment            $xx (xx ...)
```

The increment/decrement bit is 1 for increment and 0 for decrement. The equalisation bands should be ordered increasingly with reference to frequency. All frequencies don't have to be declared. Adjustments with the value $00 should be omitted. A frequency should only be described once in the frame.

1.1.25 Reverb

Yet another subjective one. You may here adjust echoes of different kinds. Reverb left/right is the delay between every bounce in ms. Reverb bounces left/right is the number of bounces that should be made. $FF equals an infinite number of bounces. Feedback is the amount of volume that should be returned to the next echo bounce. $00 is 0%, $FF is 100%. If this value were $7F, there would be 50% volume reduction on the first bounce, yet 50% on the second and so on. Left to left means the sound from the left bounce to be played in the left speaker, while left to right means sound from the left bounce to be played in the right speaker.

'Premix left to right' is the amount of left sound to be mixed in the right before any reverb is applied, where $00 id 0% and $FF is 100%. 'Premix right to left' does the same thing, but right to left. Setting both premix to $FF would result in a mono output (if the reverb is applied symmetric). There may only be one "REV" frame in each tag.

```
Reverb settings                 "REV"
Frame size                      $00 00 0C
Reverb left (ms)                $xx xx
Reverb right (ms)               $xx xx
Reverb bounces, left            $xx
Reverb bounces, right           $xx
Reverb feedback, left to left   $xx
Reverb feedback, left to right  $xx
Reverb feedback, right to right $xx
Reverb feedback, right to left  $xx
Premix left to right            $xx
Premix right to left            $xx
```

1.1.26 Attached picture

This frame contains a picture directly related to the audio file. Image format is preferably "PNG" [PNG] or "JPG" [JFIF]. Description is a short description of the picture, represented as a terminated textstring. The description has a maximum length of 64 characters, but may be empty. There may be several pictures attached to one file, each in their individual "PIC" frame, but only one with the same content descriptor. There may only be one picture with the picture type declared as picture type $01 and $02 respectively. There is a possibility to put only a link to the image file by using the 'image format' "–>" and having a complete URL [URL] instead of picture data. The use of linked files should however be used restrictively since there is the risk of separation of files.

```
Attached picture   "PIC"
Frame size         $xx xx xx
Text encoding      $xx
Image format       $xx xx xx
Picture type       $xx
Description        <textstring> $00 (00)
Picture data       <binary data>
```

```
Picture type:  $00  Other
               $01  32x32 pixels 'file icon' (PNG only)
               $02  Other file icon
               $03  Cover (front)
               $04  Cover (back)
               $05  Leaflet page
               $06  Media (e.g. lable side of CD)
               $07  Lead artist/lead performer/soloist
               $08  Artist/performer
               $09  Conductor
               $0A  Band/Orchestra
               $0B  Composer
               $0C  Lyricist/text writer
               $0D  Recording Location
               $0E  During recording
               $0F  During performance
               $10  Movie/video screen capture
               $11  A bright coloured fish
               $12  Illustration
               $13  Band/artist logotype
               $14  Publisher/Studio logotype
```

1.1.27 General encapsulated object

In this frame any type of file can be encapsulated. After the header, 'Frame size' and 'Encoding' follows 'MIME type' [MIME] and 'Filename' for the encapsulated object, both represented as terminated strings encoded with ISO 8859-1 [ISO-8859-1]. The filename is case sensitive. Then follows a content description as terminated string, encoded as 'Encoding'. The last thing in the frame is the actual object. The first two strings may be omitted, leaving only their terminations. MIME type is always an ISO-8859-1 text string. There may be more than one "GEO" frame in each tag, but only one with the same content descriptor.

```
General encapsulated object "GEO"
Frame size                  $xx xx xx
Text encoding               $xx
MIME type                   <textstring> $00
Filename                    <textstring> $00 (00)
Content description         <textstring> $00 (00)
Encapsulated object         <binary data>
```

1.1.28 Play counter

This is simply a counter of the number of times a file has been played. The value is increased by one every time the file begins to play. There may only be one "CNT" frame in each tag. When the counter reaches all one's, one byte is inserted in front of the counter thus making the counter eight bits bigger. The counter must be at least 32-bits long to begin with.

```
Play counter    "CNT"
Frame size      $xx xx xx
Counter         $xx xx xx xx (xx ...)
```

1.1.29 Popularimeter

The purpose of this frame is to specify how good an audio file is. Many interesting applications could be found to this frame such as a playlist that features better audiofiles more often than others or it could be used to profile a persons taste and find other 'good' files by comparing people's profiles. The frame is very simple. It contains the email address to the user, one rating byte and a four byte play counter, intended to be increased with one for every time the file is played. The email is a terminated string. The rating is 1-255 where 1 is worst and 255 is best. 0 is unknown. If no personal counter is wanted it may be omitted. When the counter reaches all one's, one byte is inserted in front of the counter thus making the counter eight bits bigger in the same away as the play counter ("CNT"). There may be more than one "POP" frame in each tag, but only one with the same email address.

```
Popularimeter   "POP"
Frame size      $xx xx xx
Email to user   <textstring> $00
Rating          $xx
Counter         $xx xx xx xx (xx ...)
```

1.1.30 Recommended buffer size

Sometimes the server from which a audio file is streamed is aware of transmission or coding problems resulting in interruptions in the audio stream. In these cases, the size of the buffer can be recommended by the server using this frame. If the 'embedded info flag' is true (1) then this indicates that an ID3 tag with the maximum size described in 'Buffer size' may occur in the audiostream. In such case the tag should reside between two MPEG [MPEG] frames, if the audio is MPEG encoded. If the position of the next tag is known, 'offset to next tag' may be used. The offset is calculated from the end of tag in which this frame resides to the first byte of the header in the next. This field may be omitted. Embedded tags is currently not recommended since this could render unpredictable behaviour from present software/hardware. The 'Buffer size' should be kept to a minimum. There may only be one "BUF" frame in each tag.

```
Recommended buffer size   "BUF"
Frame size                $xx xx xx
Buffer size               $xx xx xx
Embedded info flag        %0000000x
Offset to next tag        $xx xx xx xx
```

1.1.31 Encrypted meta frame

This frame contains one or more encrypted frames. This enables protection of copyrighted information such as pictures and text, that people might want to pay extra for. Since standardisation of such an encryption scheme is beyond this document, all "CRM" frames begin with a terminated string with a URL [URL] containing an email address, or a link to a location where an email adress can be found, that belongs to the organisation responsible for this specific encrypted meta frame.

Questions regarding the encrypted frame should be sent to the indicated email address. If a $00 is found directly after the 'Frame size', the whole frame should be ignored, and preferably be removed. The 'Owner identifier' is then followed by a short content description and explanation as to why it's encrypted. After the 'content/explanation' description, the actual encrypted block follows.

When an ID3v2 decoder encounters a "CRM" frame, it should send the datablock to the 'plugin' with the corresponding 'owner identifier' and expect to receive either a datablock with one or several ID3v2 frames after each other or an error. There may be more than one "CRM" frames in a tag, but only one with the same 'owner identifier'.

```
Encrypted meta frame  "CRM"
Frame size            $xx xx xx
Owner identifier      <textstring> $00 (00)
Content/explanation   <textstring> $00 (00)
Encrypted datablock   <binary data>
```

1.1.32 Audio encryption

This frame indicates if the actual audio stream is encrypted, and by whom. Since standardisation of such encrypion scheme is beyond this document, all "CRA" frames begin with a terminated string with a URL containing an email address, or a link to a location where an email address can be found, that belongs to the organisation responsible for this specific encrypted audio file. Questions regarding the encrypted audio should be sent to the email address specified. If a $00 is found directly after the 'Frame size' and the audiofile indeed is encrypted, the whole file may be considered useless.

After the 'Owner identifier', a pointer to an unencrypted part of the audio can be specified. The 'Preview start' and 'Preview length' is described in frames. If no part is unencrypted, these fields should be left zeroed. After the 'preview length' field follows optionally a datablock required for decryption of the audio. There may be more than one "CRA" frames in a tag, but only one with the same 'Owner identifier'.

```
Audio encryption   "CRA"
Frame size         $xx xx xx
Owner identifier   <textstring> $00 (00)
Preview start      $xx xx
Preview length     $xx xx
Encryption info    <binary data>
```

1.1.33 Linked information

To keep space waste as low as possible this frame may be used to link information from another ID3v2 tag that might reside in another audio file or alone in a binary file. It is recommended that this method is only used when the files are stored on a CD-ROM or other circumstances when the risk of file seperation is low. The frame contains a frame identifier, which is the frame that should be linked into this tag, a URL [URL] field, where a reference to the file where the frame is given, and additional ID data, if needed. Data should be retrieved from the first tag found in the file to which this link points. There may be more than one "LNK" frame in a tag, but only one with the same contents. A linked frame is to be considered as part of the tag and has the same restrictions as if it was a physical part of the tag (i.e. only one "REV" frame allowed, whether it's linked or not).

```
Linked information  "LNK"
Frame size          $xx xx xx
Frame identifier    $xx xx xx
URL                 <textstring> $00 (00)
Additional ID data  <textstring(s)>
```

Frames that may be linked and need no additional data are "IPL", "MCI", "ETC", "LLT", "STC", "RVA", "EQU", "REV", "BUF", the text information frames and the URL link frames.

The "TXX", "PIC", "GEO", "CRM" and "CRA" frames may be linked with the content descriptor as additional ID data.

The "COM", "SLT" and "ULT" frames may be linked with three bytes of language descriptor directly followed by a content descriptor as additional ID data.

1.1.34 The 'unsynchronisation scheme'

The only purpose of the 'unsychronisation scheme' is to make the ID3v2 tag as compatible as possible with existing software. There is no use in 'unsynchronising' tags if the file is only to be processed by new software. Unsynchronisation may only be made with MPEG 2 layer I, II and III and MPEG 2.5 files.

Whenever a false synchronisation is found within the tag, one zeroed byte is inserted after the first false synchronisation byte. The format of a correct sync 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 won't be affected by the decoding process. Therefore all the $FF 00 combinations have to be replaced with the $FF 00 00 combination during the unsynchonisation.

To indicate usage of the unsynchronisation, the first bit in 'ID3 flags' should be set. This bit should only be set if the tag contained a, now corrected, false synchronisation. The bit should only be clear if the tag does not contain any false synchronisations.

Do bear in mind, that if a compression scheme is used by the encoder, the unsyncronisation scheme should be applied *afterwards*. When decoding a compressed, 'unsyncronised' file, the 'unsyncronisation scheme' should be parsed first, compression afterwards.

1.1.35 Copyright

1.1.36 References

[CDDB] Compact Disc Data Base

<url:http://www.cddb.com>

[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-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

[ISRC] ISO 3901:1986 International Standard Recording Code (ISRC). Technical committee / subcommittee: TC 46 / SC 9

[JFIF] JPEG File Interchange Format, version 1.02

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

[MIME] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, November 1996.

<url:ftp://ftp.isi.edu/in-notes/rfc2045.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] ISO/IEC 10646-1:1993. Universal Multiple-Octet Coded Character Set (UCS), Part 1: Architecture and Basic Multilingual Plane. Technical committee / subcommittee: JTC 1 / SC 2

<url:http://www.unicode.org>

[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>

1.1.37 Appendix

Appendix A - ID3-Tag Specification V1.1

ID3-Tag Specification V1.1 (12 dec 1997) by Michael Mutschler <amiga2@info2.rus.uni-stuttgart.de>, edited for space and clarity reasons.

Overview

The ID3-Tag is an information field for MPEG Layer 3 audio files. Since a standalone MP3 doesn't provide a method of storing other information than those directly needed for replay reasons, the ID3-tag was invented by Eric Kemp in 1996.

A revision from ID3v1 to ID3v1.1 was made by Michael Mutschler to support track number information is described in A.4.

ID3v1 Implementation

The Information is stored in the last 128 bytes of an MP3. The Tag has got the following fields, and the offsets given here, are from 0-127.

> Field Length Offsets Tag 3 0-2 Songname 30 3-32 Artist 30 33-62 Album 30 63-92 Year 4 93-96 Comment 30 97-126 Genre 1 127

The string-fields contain ASCII-data, coded in ISO-Latin 1 codepage. Strings which are smaller than the field length are padded with zero- bytes.

> **Tag: The tag is valid if this field contains the string "TAG". This** has to be uppercase!

Songname: This field contains the title of the MP3 (string as above).

Artist: This field contains the artist of the MP3 (string as above).

Album: this field contains the album where the MP3 comes from (string as above).

Year: this field contains the year when this song has originally been released (string as above).

Comment: this field contains a comment for the MP3 (string as above). Revision to this field has been made in ID3v1.1. See A.4.

Genre: this byte contains the offset of a genre in a predefined list the byte is treated as an unsigned byte. The offset is starting from 0. See A.3.

Genre List

The following genres is defined in ID3v1

0.Blues 1.Classic Rock 2.Country 3.Dance 4.Disco 5.Funk 6.Grunge 7.Hip-Hop 8.Jazz 9.Metal 10.New Age 11.Oldies 12.Other 13.Pop 14.R&B 15.Rap 16.Reggae 17.Rock 18.Techno 19.Industrial 20.Alternative 21.Ska 22.Death Metal 23.Pranks 24.Soundtrack 25.Euro-Techno 26.Ambient 27.Trip-Hop 28.Vocal 29.Jazz+Funk 30.Fusion 31.Trance 32.Classical 33.Instrumental 34.Acid 35.House 36.Game 37.Sound Clip 38.Gospel 39.Noise 40.AlternRock 41.Bass 42.Soul 43.Punk 44.Space 45.Meditative 46.Instrumental Pop 47.Instrumental Rock 48.Ethnic 49.Gothic 50.Darkwave 51.Techno-Industrial 52.Electronic 53.Pop-Folk 54.Eurodance 55.Dream 56.Southern Rock 57.Comedy 58.Cult 59.Gangsta 60.Top 40 61.Christian Rap 62.Pop/Funk 63.Jungle 64.Native American 65.Cabaret 66.New Wave 67.Psychadelic 68.Rave 69.Showtunes 70.Trailer 71.Lo-Fi 72.Tribal 73.Acid Punk 74.Acid Jazz 75.Polka 76.Retro 77.Musical 78.Rock & Roll 79.Hard Rock

The following genres are Winamp extensions

> 80.Folk 81.Folk-Rock 82.National Folk 83.Swing 84.Fast Fusion 85.Bebob 86.Latin 87.Revival 88.Celtic 89.Bluegrass 90.Avantgarde 91.Gothic Rock 92.Progressive Rock 93.Psychedelic Rock 94.Symphonic Rock 95.Slow Rock 96.Big Band 97.Chorus 98.Easy Listening 99.Acoustic

100.Humour 101.Speech 102.Chanson 103.Opera 104.Chamber Music 105.Sonata 106.Symphony 107.Booty Bass 108.Primus 109.Porn Groove 110.Satire 111.Slow Jam 112.Club 113.Tango 114.Samba 115.Folklore 116.Ballad 117.Power Ballad 118.Rhythmic Soul 119.Freestyle 120.Duet 121.Punk Rock 122.Drum Solo 123.A capella 124.Euro-House 125.Dance Hall

Track addition - ID3v1.1

> In ID3v1.1, Michael Mutschler revised the specification of the comment field in order to implement the track number. The new format of the comment field is a 28 character string followed by a mandatory null ($00) character and the original album tracknumber stored as an unsigned byte-size integer. In such cases where the 29th byte is not the null character or when the 30th is a null character, the tracknumber is to be considered undefined.

1.1.38 Author's Address


```
Martin Nilsson
Rydsvägen 246 C. 30
S-584 34 Linköping
Sweden

Email: nilsson at id3.org
```

```
Co-authors:

Johan Sundström   Email: johan at id3.org
```

1.2 ID3 tag version 2.3.0

Status of this document

This document is an informal standard and replaces the ID3v2.2.0 standard [ID3v2]. The informal standard is released so that implementors could have a set standard before a formal standard is set. The formal standard will use another version or revision number if not identical to what is described in this 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 ID3v2.3.0, which is a more developed version of the ID3v2 informal standard [ID3v2] (version 2.2.0), evolved from the ID3 tagging system. The ID3v2 offers a flexible way of storing information about an audio file within itself to determine its origin and contents. The information may be technical information, such as equalisation curves, as well as related meta information, such as title, performer, copyright etc.

1. Table of contents

9. Author's Address

2. Conventions in this document

   In the examples, text within "" is a text string exactly as it appears in a file. 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 0-9 only.

3. ID3v2 overview

   The two biggest design goals were to be able to implement ID3v2 without disturbing old software too much and that ID3v2 should be as flexible and expandable as possible.

   The first criterion is met by the simple fact that the MPEG [MPEG] decoding software uses a syncsignal, embedded in the audiostream, to 'lock on to' the audio. Since the ID3v2 tag doesn't contain a valid syncsignal, no software will attempt to play the tag. If, for any reason, coincidence make a syncsignal appear within the tag it will be taken care of by the 'unsynchronisation scheme' described in section 5.

   The second criterion has made a more noticeable impact on the design of the ID3v2 tag. It 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 there is an identifier that explains the frames' format and content, and a size descriptor that allows software to skip unknown frames.

   If a total revision of the ID3v2 tag should be needed, there is a version number and a size descriptor in the ID3v2 header.

   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.

   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).

   It is permitted to include padding after all 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 head of the tag. 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.

3.1. ID3v2 header

The ID3v2 tag header, which should be the first information in the file, is 10 bytes as follows:

> ID3v2/file identifier "ID3" ID3v2 version $03 00 ID3v2 flags %abc00000 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 it's major version, while the second byte is its revision number. In this case this is ID3v2.3.0. All revisions are backwards compatible while major versions are not. If software with ID3v2.2.0 and below support should encounter version three or higher it should simply ignore the whole tag. Version and revision will never be $FF.

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

a - Unsynchronisation

> Bit 7 in the 'ID3v2 flags' indicates whether or not unsynchronisation is used (see section 5 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.

c - Experimental indicator

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

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

The ID3v2 tag size is encoded with four bytes where the most significant bit (bit 7) is set to zero in every byte, making a total of 28 bits. The zeroed bits are ignored, so a 257 bytes long tag is represented as $00 00 02 01.

The ID3v2 tag size is the size of the complete tag after unsychronisation, including padding, excluding the header but not excluding the extended header (total tag size - 10). Only 28 bits (representing up to 256MB) are used in the size description to avoid the introducuction of 'false syncsignals'.

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. ID3v2 extended header

The extended header contains information that is not vital to the correct parsing of the tag information, hence the extended header is optional.

Extended header size $xx xx xx xx Extended Flags $xx xx Size of padding $xx xx xx xx

Where the 'Extended header size', currently 6 or 10 bytes, excludes itself. The 'Size of padding' is simply the total tag size excluding the frames and the headers, in other words the padding. The extended header is considered separate from the header proper, and as such is subject to unsynchronisation.

The extended flags are a secondary flag set which describes further attributes of the tag. These attributes are currently defined as follows

%x0000000 00000000

x - CRC data present

If this flag is set four bytes of CRC-32 data is appended to the extended header. The CRC should be calculated before unsynchronisation on the data between the extended header and the padding, i.e. the frames and only the frames.

Total frame CRC $xx xx xx xx

3.3. ID3v2 frame overview

As the tag consists of a tag header and a tag body with one or more frames, all the frames consists of a frame header followed by one or more fields containing the actual information. The layout of the frame header:

Frame ID $xx xx xx xx (four characters) Size $xx xx xx xx Flags $xx xx

The frame ID made out of the characters capital A-Z and 0-9. Identifiers beginning with "X", "Y" and "Z" are for experimental use and free for everyone to use, without the need to set the experimental bit in the tag header. Have 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, making a total header size of ten bytes in every frame. The size is calculated as frame size excluding frame header (frame size - 10).

In the frame header the size descriptor is followed by two flags bytes. These flags are described in section 3.3.1.

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 a string is represented as ISO-8859-1 [ISO-8859-1] characters in the range $20 - $FF. Such strings are represented as <text string>, or <full text string> if newlines are allowed, in the frame descriptions. All Unicode strings [UNICODE] use 16-bit unicode 2.0 (ISO/IEC 10646-1:1993, UCS-2). Unicode strings must begin with the Unicode BOM ($FF FE or $FE FF) to identify the byte order.

All numeric strings and URLs [URL] are always encoded as ISO-8859-1. Terminated strings are terminated with $00 if encoded with ISO-8859-1 and $00 00 if encoded as unicode. If nothing else is said newline character is forbidden. In ISO-8859-1 a new line is represented, when allowed, with $0A only. Frames that allow different types of text encoding have a text encoding description byte directly after the frame size. If ISO-8859-1 is used this byte should be $00, if Unicode is used it should be $01. Strings dependent on encoding is represented as <text string according to encoding>, or <full text string according to encoding> if newlines are allowed. Any empty Unicode strings which are NULL-terminated may have the Unicode BOM followed by a Unicode NULL ($FF FE 00 00 or $FE FF 00 00).

The three byte language field is used to describe the language of the frame's content, according to ISO-639-2 [ISO-639-2].

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.

3.3.1. Frame header flags

In the frame header the size descriptor is followed by two flags bytes. All unused flags must be cleared. The first byte is for 'status messages' and the second byte is for encoding purposes. If an unknown flag is set in the first byte the frame may not be changed without the bit cleared. If an unknown flag is set in the second byte it is likely to not be readable. The flags field is defined as follows.

%abc00000 %ijk00000

a - Tag alter preservation

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

i - Compression

This flag indicates whether or not the frame is compressed.

0 Frame is not compressed. 1 Frame is compressed using zlib [zlib] with 4 bytes for

'decompressed size' appended to the frame header.

j - Encryption

This flag indicates wether or not the frame is enrypted. If set one byte indicating with which method it was encrypted will be appended to the frame header. See section 4.26. for more information about encryption method registration.

0 Frame is not encrypted. 1 Frame is encrypted.

k - 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 header. Every frame with the same group identifier belongs to the same group.

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

Some flags indicates that the frame header is extended with additional information. This information will be added to the frame header in the same order as the flags indicating the additions. I.e. the four bytes of decompressed size will preceed the encryption method byte. These additions to the frame header, while not included in the frame header size but are included in the 'frame size' field, are not subject to encryption or compression.

3.3.2. Default flags

The default settings for the frames described in this document can be divided into the following classes. The flags may be set differently if found more suitable by the software.

1. Discarded if tag is altered, discarded if file is altered.

   None.

2. Discarded if tag is altered, preserved if file is altered.

   None.

3. Preserved if tag is altered, discarded if file is altered.

   AENC, ETCO, EQUA, MLLT, POSS, SYLT, SYTC, RVAD, TENC, TLEN, TSIZ

4. Preserved if tag is altered, preserved if file is altered.

   The rest of the frames.

4. Declared ID3v2 frames

   The following frames are declared in this draft.

   4.21 AENC Audio encryption 4.15 APIC Attached picture

   4.11 COMM Comments 4.25 COMR Commercial frame

   4.26 ENCR Encryption method registration 4.13 EQUA Equalization 4.6 ETCO Event timing codes

   4.16 GEOB General encapsulated object 4.27 GRID Group identification registration

   4.4 IPLS Involved people list

   4.21 LINK Linked information

   4.5 MCDI Music CD identifier 4.7 MLLT MPEG location lookup table

   4.24 OWNE Ownership frame

4.28. PRIV Private frame 4.17 PCNT Play counter 4.18 POPM Popularimeter 4.22 POSS Position synchronisation frame

4.19 RBUF Recommended buffer size 4.12 RVAD Relative volume adjustment 4.14 RVRB Reverb

4.10 SYLT Synchronized lyric/text 4.8 SYTC Synchronized tempo codes

4.2.1 TALB Album/Movie/Show title 4.2.1 TBPM BPM (beats per minute) 4.2.1 TCOM Composer 4.2.1 TCON Content type 4.2.1 TCOP Copyright message 4.2.1 TDAT Date 4.2.1 TDLY Playlist delay 4.2.1 TENC Encoded by 4.2.1 TEXT Lyricist/Text writer 4.2.1 TFLT File type 4.2.1 TIME Time 4.2.1 TIT1 Content group description 4.2.1 TIT2 Title/songname/content description 4.2.1 TIT3 Subtitle/Description refinement 4.2.1 TKEY Initial key 4.2.1 TLAN Language(s) 4.2.1 TLEN Length 4.2.1 TMED Media type 4.2.1 TOAL Original album/movie/show title 4.2.1 TOFN Original filename 4.2.1 TOLY Original lyricist(s)/text writer(s) 4.2.1 TOPE Original artist(s)/performer(s) 4.2.1 TORY Original release year 4.2.1 TOWN File owner/licensee 4.2.1 TPE1 Lead performer(s)/Soloist(s) 4.2.1 TPE2 Band/orchestra/accompaniment 4.2.1 TPE3 Conductor/performer refinement 4.2.1 TPE4 Interpreted, remixed, or otherwise modified by 4.2.1 TPOS Part of a set 4.2.1 TPUB Publisher 4.2.1 TRCK Track number/Position in set 4.2.1 TRDA Recording dates 4.2.1 TRSN Internet radio station name 4.2.1 TRSO Internet radio station owner 4.2.1 TSIZ Size 4.2.1 TSRC ISRC (international standard recording code) 4.2.1 TSSE Software/Hardware and settings used for encoding 4.2.1 TYER Year 4.2.2 TXXX User defined text information frame

4.1 UFID Unique file identifier 4.23 USER Terms of use 4.9 USLT Unsychronized lyric/text transcription

4.3.1 WCOM Commercial information 4.3.1 WCOP Copyright/Legal information 4.3.1 WOAF Official audio file webpage 4.3.1 WOAR Official artist/performer webpage 4.3.1 WOAS Official audio source webpage 4.3.1 WORS Official internet radio station homepage 4.3.1 WPAY Payment 4.3.1 WPUB Publishers official webpage 4.3.2 WXXX User defined URL link frame

4.1. Unique file identifier

This frame's purpose is to be able to identify the audio file in a database that may contain more information relevant to the content. Since standardisation of such a database is beyond this document, all frames begin with a null-terminated string with a URL [URL] containing an email address, or a link to a location where an email address can be found, that belongs to the organisation responsible for this specific database implementation. Questions regarding the database should be sent to the indicated email address. The URL should not be used for the actual database queries. The string "http://www.id3.org/dummy/ufid.html" should be used for tests. Software that isn't told otherwise may safely remove such frames. The 'Owner identifier' must be non-empty (more than just a termination). The 'Owner identifier' is then followed by the actual identifier, which may be up to 64 bytes. There may be more than one "UFID" frame in a tag, but only one with the same 'Owner identifier'.

<Header for 'Unique file identifier', ID: "UFID"> Owner identifier <text string> $00 Identifier <up to 64 bytes binary data>

4.2. Text information frames

The text information frames are the most important frames, containing information like artist, album and more. There may only be one text information frame of its kind in an tag. If the textstring is followed by a termination ($00 (00)) all the following information should be ignored and not be displayed. All text frame identifiers begin with "T". Only text frame identifiers begin with "T", with the exception of the "TXXX" frame. All the text information frames have the following format:

<Header for 'Text information frame', ID: "T000" - "TZZZ", excluding "TXXX" described in 4.2.2.> Text encoding $xx Information <text string according to encoding>

4.2.1. Text information frames - details

TALB The 'Album/Movie/Show title' frame is intended for the title of the recording(/source of sound) which the audio in the file is taken from.

TBPM The 'BPM' frame contains the number of beats per minute in the mainpart of the audio. The BPM is an integer and represented as a numerical string.

TCOM The 'Composer(s)' frame is intended for the name of the composer(s). They are seperated with the "/" character.

TCON The 'Content type', which previously was stored as a one byte numeric value only, is now a numeric string. You may use one or several of the types as ID3v1.1 did or, since the category list would be impossible to maintain with accurate and up to date categories, define your own.

References to the ID3v1 genres can be made by, as first byte, enter "(" followed by a number from the genres list (appendix A.) and ended with a ")" character. This is optionally followed by a refinement, e.g. "(21)" or "(4)Eurodisco". Several references can be made in the same frame, e.g. "(51)(39)". If the refinement should begin with a "(" character it should be replaced with "((", e.g. "((I can figure out any genre)" or "(55)((I think...)". The following new content types is defined in ID3v2 and is implemented in the same way as the numerig content types, e.g. "(RX)".

RX Remix CR Cover

TCOP The 'Copyright message' frame, which must begin with a year and a space character (making five characters), is intended for the copyright holder of the original sound, not the audio file itself. The absence of this frame means only that the copyright information is unavailable or has been removed, and must not be interpreted to mean that the sound is public domain. Every time this field is displayed the field must be preceded with "Copyright " (C) " ", where (C) is one character showing a C in a circle.

TDAT The 'Date' frame is a numeric string in the DDMM format containing the date for the recording. This field is always four characters long.

TDLY The 'Playlist delay' defines the numbers of milliseconds of silence between every song in a playlist. The player should use the "ETC" frame, if present, to skip initial silence and silence at the end of the audio to match the 'Playlist delay' time. The time is represented as a numeric string.

TENC The 'Encoded by' frame contains the name of the person or organisation that encoded the audio file. This field may contain a copyright message, if the audio file also is copyrighted by the encoder.

TEXT The 'Lyricist(s)/Text writer(s)' frame is intended for the writer(s) of the text or lyrics in the recording. They are seperated with the "/" character.

TFLT The 'File type' frame indicates which type of audio this tag defines. The following type and refinements are defined:

MPG MPEG Audio

/1	MPEG 1/2 layer I
/2	MPEG 1/2 layer II
/3	MPEG 1/2 layer III

/2.5 MPEG 2.5 /AAC Advanced audio compression

VQF Transform-domain Weighted Interleave Vector Quantization PCM Pulse Code Modulated audio

but other types may be used, not for these types though. This is used in a similar way to the predefined types in the "TMED" frame, but without parentheses. If this frame is not present audio type is assumed to be "MPG".

TIME The 'Time' frame is a numeric string in the HHMM format containing the time for the recording. This field is always four characters long.

TIT1 The 'Content group description' frame is used if the sound belongs to a larger category of sounds/music. For example, classical music is often sorted in different musical sections (e.g. "Piano Concerto", "Weather - Hurricane").

TIT2 The 'Title/Songname/Content description' frame is the actual name of the piece (e.g. "Adagio", "Hurricane Donna").

TIT3 The 'Subtitle/Description refinement' frame is used for information directly related to the contents title (e.g. "Op. 16" or "Performed live at Wembley").

TKEY The 'Initial key' frame contains the musical key in which the sound starts. It is represented as a string with a maximum length of three characters. The ground keys are represented with "A","B","C","D","E", "F" and "G" and halfkeys represented with "b" and "#". Minor is represented as "m". Example "Cbm". Off key is represented with an "o" only.

TLAN The 'Language(s)' frame should contain the languages of the text or lyrics spoken or sung in the audio. The language is represented with three characters according to ISO-639-2. If more than one language is used in the text their language codes should follow according to their usage.

TLEN The 'Length' frame contains the length of the audiofile in milliseconds, represented as a numeric string.

TMED The 'Media type' frame describes from which media the sound originated. This may be a text string or a reference to the predefined media types found in the list below. References are made within "(" and ")" and are optionally followed by a text refinement, e.g. "(MC) with four channels". If a text refinement should begin with a "(" character it should be replaced with "((" in the same way as in the "TCO" frame. Predefined refinements is appended after the media type, e.g. "(CD/A)" or "(VID/PAL/VHS)".

DIG Other digital media

/A	Analog transfer from media

ANA Other analog media

/WAC	Wax cylinder
/8CA	8-track tape cassette

CD CD

/A	Analog transfer from media
/DD	DDD
/AD	ADD
/AA	AAD

LD Laserdisc

/A	Analog transfer from media

TT Turntable records

/33	33.33 rpm
/45	45 rpm
/71	71.29 rpm
/76	76.59 rpm

/78	78.26 rpm
/80	80 rpm
MD MiniDisc	
/A	Analog transfer from media
DAT DAT	
/A	Analog transfer from media
/1	standard, 48 kHz/16 bits, linear
/2	mode 2, 32 kHz/16 bits, linear
/3	mode 3, 32 kHz/12 bits, nonlinear, low speed
/4	mode 4, 32 kHz/12 bits, 4 channels
/5	mode 5, 44.1 kHz/16 bits, linear
/6	mode 6, 44.1 kHz/16 bits, 'wide track' play
DCC DCC	
/A	Analog transfer from media
DVD DVD	
/A	Analog transfer from media
TV Television	
/PAL	PAL
/NTSC	NTSC
/SECAM	SECAM
VID Video	
/PAL	PAL
/NTSC	NTSC
/SECAM	SECAM
/VHS	VHS
/SVHS	S-VHS
/BETA	BETAMAX
RAD Radio	
/FM	FM
/AM	AM
/LW	LW
/MW	MW
TEL Telephone	
/I	ISDN
MC MC (normal cassette)	
/4	4.75 cm/s (normal speed for a two sided cassette)

/9	9.5 cm/s
/I	Type I cassette (ferric/normal)
/II	Type II cassette (chrome)
/III	Type III cassette (ferric chrome)
/IV	Type IV cassette (metal)

REE Reel

/9	9.5 cm/s
/19	19 cm/s
/38	38 cm/s
/76	76 cm/s
/I	Type I cassette (ferric/normal)
/II	Type II cassette (chrome)
/III	Type III cassette (ferric chrome)
/IV	Type IV cassette (metal)

TOAL The 'Original album/movie/show title' frame is intended for the title of the original recording (or source of sound), if for example the music in the file should be a cover of a previously released song.

TOFN The 'Original filename' frame contains the preferred filename for the file, since some media doesn't allow the desired length of the filename. The filename is case sensitive and includes its suffix.

TOLY The 'Original lyricist(s)/text writer(s)' frame is intended for the text writer(s) of the original recording, if for example the music in the file should be a cover of a previously released song. The text writers are seperated with the "/" character.

TOPE The 'Original artist(s)/performer(s)' frame is intended for the performer(s) of the original recording, if for example the music in the file should be a cover of a previously released song. The performers are seperated with the "/" character.

TORY The 'Original release year' frame is intended for the year when the original recording, if for example the music in the file should be a cover of a previously released song, was released. The field is formatted as in the "TYER" frame.

TOWN The 'File owner/licensee' frame contains the name of the owner or licensee of the file and it's contents.

TPE1 The 'Lead artist(s)/Lead performer(s)/Soloist(s)/Performing group' is used for the main artist(s). They are seperated with the "/" character.

TPE2 The 'Band/Orchestra/Accompaniment' frame is used for additional information about the performers in the recording.

TPE3 The 'Conductor' frame is used for the name of the conductor.

TPE4 The 'Interpreted, remixed, or otherwise modified by' frame contains more information about the people behind a remix and similar interpretations of another existing piece.

TPOS The 'Part of a set' frame is a numeric string that describes which part of a set the audio came from. This frame is used if the source described in the "TALB" frame is divided into several mediums, e.g. a double CD. The value may be extended with a "/" character and a numeric string containing the total number of parts in the set. E.g. "1/2".

TPUB The 'Publisher' frame simply contains the name of the label or publisher.

TRCK The 'Track number/Position in set' frame is a numeric string containing the order number of the audio-file on its original recording. This may be extended with a "/" character and a numeric string containing the total numer of tracks/elements on the original recording. E.g. "4/9".

TRDA The 'Recording dates' frame is a intended to be used as complement to the "TYER", "TDAT" and "TIME" frames. E.g. "4th-7th June, 12th June" in combination with the "TYER" frame.

TRSN The 'Internet radio station name' frame contains the name of the internet radio station from which the audio is streamed.

TRSO The 'Internet radio station owner' frame contains the name of the owner of the internet radio station from which the audio is streamed.

TSIZ The 'Size' frame contains the size of the audiofile in bytes, excluding the ID3v2 tag, represented as a numeric string.

TSRC The 'ISRC' frame should contain the International Standard Recording Code [ISRC] (12 characters).

TSSE The 'Software/Hardware and settings used for encoding' frame includes the used audio encoder and its settings when the file was encoded. Hardware refers to hardware encoders, not the computer on which a program was run.

TYER The 'Year' frame is a numeric string with a year of the recording. This frames is always four characters long (until the year 10000).

4.2.2. User defined text information frame

This frame is intended for one-string text information concerning the audiofile in a similar way to the other "T"-frames. The frame body consists of a description of the string, represented as a terminated string, followed by the actual string. There may be more than one "TXXX" frame in each tag, but only one with the same description.

<Header for 'User defined text information frame', ID: "TXXX"> Text encoding $xx Description <text string according to encoding> $00 (00) Value <text string according to encoding>

4.3. URL link frames

With these frames dynamic data such as webpages with touring information, price information or plain ordinary news can be added to the tag. There may only be one URL [URL] link frame of its kind in an tag, except when stated otherwise in the frame description. If the textstring is followed by a termination ($00 (00)) all the following information should be ignored and not be displayed. All URL link frame identifiers begins with "W". Only URL link frame identifiers begins with "W". All URL link frames have the following format:

<Header for 'URL link frame', ID: "W000" - "WZZZ", excluding "WXXX" described in 4.3.2.> URL <text string>

4.3.1. URL link frames - details

WCOM The 'Commercial information' frame is a URL pointing at a webpage with information such as where the album can be bought. There may be more than one "WCOM" frame in a tag, but not with the same content.

WCOP The 'Copyright/Legal information' frame is a URL pointing at a webpage where the terms of use and ownership of the file is described.

WOAF The 'Official audio file webpage' frame is a URL pointing at a file specific webpage.

WOAR The 'Official artist/performer webpage' frame is a URL pointing at the artists official webpage. There may be more than one "WOAR" frame in a tag if the audio contains more than one performer, but not with the same content.

WOAS The 'Official audio source webpage' frame is a URL pointing at the official webpage for the source of the audio file, e.g. a movie.

WORS The 'Official internet radio station homepage' contains a URL pointing at the homepage of the internet radio station.

WPAY The 'Payment' frame is a URL pointing at a webpage that will handle the process of paying for this file.

WPUB The 'Publishers official webpage' frame is a URL pointing at the official wepage for the publisher.

4.3.2. User defined URL link frame

This frame is intended for URL [URL] links concerning the audiofile in a similar way to the other "W"-frames. The frame body consists of a description of the string, represented as a terminated string, followed by the actual URL. The URL is always encoded with ISO-8859-1 [ISO-8859-1]. There may be more than one "WXXX" frame in each tag, but only one with the same description.

<Header for 'User defined URL link frame', ID: "WXXX"> Text encoding $xx Description <text string according to encoding> $00 (00) URL <text string>

4.4. Involved people list

Since there might be a lot of people contributing to an audio file in various ways, such as musicians and technicians, the 'Text information frames' are often insufficient to list everyone involved in a project. The 'Involved people list' is a frame containing the names of those involved, and how they were involved. The body simply contains a terminated string with the involvement directly followed by a terminated string with the involvee followed by a new involvement and so on. There may only be one "IPLS" frame in each tag.

<Header for 'Involved people list', ID: "IPLS"> Text encoding $xx People list strings <text strings according to encoding>

4.5. Music CD identifier

This frame is intended for music that comes from a CD, so that the CD can be identified in databases such as the CDDB [CDDB]. The frame consists of a binary dump of the Table Of Contents, TOC, from the CD, which is a header of 4 bytes and then 8 bytes/track on the CD plus 8 bytes for the 'lead out' making a maximum of 804 bytes. The offset to the beginning of every track on the CD should be described with a four bytes absolute CD-frame address per track, and not with absolute time. This frame requires a present and valid "TRCK" frame, even if the CD's only got one track. There may only be one "MCDI" frame in each tag.

<Header for 'Music CD identifier', ID: "MCDI"> CD TOC <binary data>

4.6. Event timing codes

This frame allows synchronisation with key events in a song or sound. The header is:

<Header for 'Event timing codes', ID: "ETCO"> Time stamp format $xx

Where time stamp format is:

$01 Absolute time, 32 bit sized, using MPEG [MPEG] frames as unit $02 Absolute time, 32 bit sized, using milliseconds as unit

Abolute time means that every stamp contains the time from the beginning of the file.

Followed by a list of key events in the following format:

Type of event $xx Time stamp $xx (xx ...)

The 'Time stamp' is set to zero if directly at the beginning of the sound or after the previous event. All events should be sorted in chronological order. The type of event is as follows:

$00 padding (has no meaning) $01 end of initial silence $02 intro start $03 mainpart start $04 outro start $05 outro end $06 verse start $07 refrain start $08 interlude start $09 theme start $0A variation start $0B key change $0C time change $0D momentary unwanted noise (Snap, Crackle & Pop) $0E sustained noise $0F sustained noise end $10 intro end $11 mainpart end $12 verse end $13 refrain end $14 theme end

$15-$DF reserved for future use

$E0-$EF not predefined sync 0-F

$F0-$FC reserved for future use

$FD audio end (start of silence) $FE audio file ends $FF one more byte of events follows (all the following bytes with

the value $FF have the same function)

Terminating the start events such as "intro start" is not required. The 'Not predefined sync's ($E0-EF) are for user events. You might want to synchronise your music to something, like setting of an explosion on-stage, turning on your screensaver etc.

There may only be one "ETCO" frame in each tag.

4.7. MPEG location lookup table

To increase performance and accuracy of jumps within a MPEG [MPEG] audio file, frames with time-codes in different locations in the file might be useful. The ID3v2 frame includes references that the software can use to calculate positions in the file. After the frame header is a descriptor of how much the 'frame counter' should increase for every reference. If this value is two then the first reference points out the second frame, the 2nd reference the 4th frame, the 3rd reference the 6th frame etc. In a similar way the 'bytes between reference' and 'milliseconds between reference' points out bytes and milliseconds respectively.

Each reference consists of two parts; a certain number of bits, as defined in 'bits for bytes deviation', that describes the difference between what is said in 'bytes between reference' and the reality and a certain number of bits, as defined in 'bits for milliseconds deviation', that describes the difference between what is said in 'milliseconds between reference' and the reality. The number of bits in every reference, i.e. 'bits for bytes deviation'+'bits for milliseconds deviation', must be a multiple of four. There may only be one "MLLT" frame in each tag.

<Header for 'Location lookup table', ID: "MLLT"> MPEG frames between reference $xx xx Bytes between reference $xx xx xx Milliseconds between reference $xx xx xx Bits for bytes deviation $xx Bits for milliseconds dev. $xx

Then for every reference the following data is included;

Deviation in bytes %xxx.... Deviation in milliseconds %xxx....

4.8. Synchronised tempo codes

For a more accurate description of the tempo of a musical piece this frame might be used. After the header follows one byte describing which time stamp format should be used. Then follows one or more tempo codes. Each tempo code consists of one tempo part and one time part. The tempo is in BPM described with one or two bytes. If the first byte has the value $FF, one more byte follows, which is added to the first giving a range from 2 - 510 BPM, since $00 and $01 is reserved. $00 is used to describe a beat-free time period, which is not the same as a music-free time period. $01 is used to indicate one single beat-stroke followed by a beat-free period.

The tempo descriptor is followed by a time stamp. Every time the tempo in the music changes, a tempo descriptor may indicate this for the player. All tempo descriptors should be sorted in chronological order. The first beat-stroke in a time-period is at the same time as the beat description occurs. There may only be one "SYTC" frame in each tag.

<Header for 'Synchronised tempo codes', ID: "SYTC"> Time stamp format $xx Tempo data <binary data>

Where time stamp format is:

$01 Absolute time, 32 bit sized, using MPEG [MPEG] frames as unit $02 Absolute time, 32 bit sized, using milliseconds as unit

Abolute time means that every stamp contains the time from the beginning of the file.

4.9. Unsychronised lyrics/text transcription

This frame contains the lyrics of the song or a text transcription of other vocal activities. The head includes an encoding descriptor and a content descriptor. The body consists of the actual text. The 'Content descriptor' is a terminated string. If no descriptor is entered, 'Content descriptor' is $00 (00) only. Newline characters are allowed in the text. There may be more than one 'Unsynchronised lyrics/text transcription' frame in each tag, but only one with the same language and content descriptor.

<Header for 'Unsynchronised lyrics/text transcription', ID: "USLT"> Text encoding $xx Language $xx xx xx Content descriptor <text string according to encoding> $00 (00) Lyrics/text <full text string according to encoding>

4.10. Synchronised lyrics/text

This is another way of incorporating the words, said or sung lyrics, in the audio file as text, this time, however, in sync with the audio. It might also be used to describing events e.g. occurring on a stage or on the screen in sync with the audio. The header includes a content descriptor, represented with as terminated textstring. If no descriptor is entered, 'Content descriptor' is $00 (00) only.

<Header for 'Synchronised lyrics/text', ID: "SYLT"> Text encoding $xx Language $xx xx xx Time stamp format $xx Content type $xx Content descriptor <text string according to encoding> $00 (00)

Encoding: $00 ISO-8859-1 [ISO-8859-1] character set is used => $00

is sync identifier.

$01 Unicode [UNICODE] character set is used => $00 00 is sync identifier.

Content type: $00 is other $01 is lyrics $02 is text transcription $03 is movement/part name (e.g. "Adagio") $04 is events (e.g. "Don Quijote enters the stage") $05 is chord (e.g. "Bb F Fsus") $06 is trivia/'pop up' information

Time stamp format is:

$01 Absolute time, 32 bit sized, using MPEG [MPEG] frames as unit $02 Absolute time, 32 bit sized, using milliseconds as unit

Abolute time means that every stamp contains the time from the beginning of the file.

The text that follows the frame header differs from that of the unsynchronised lyrics/text transcription in one major way. Each syllable (or whatever size of text is considered to be convenient by the encoder) is a null terminated string followed by a time stamp denoting where in the sound file it belongs. Each sync thus has the following structure:

Terminated text to be synced (typically a syllable) Sync identifier (terminator to above string) $00 (00) Time stamp $xx (xx ...)

The 'time stamp' is set to zero or the whole sync is omitted if located directly at the beginning of the sound. All time stamps should be sorted in chronological order. The sync can be considered as a validator of the subsequent string.

Newline ($0A) characters are allowed in all "SYLT" frames and should be used after every entry (name, event etc.) in a frame with the content type $03 - $04.

A few considerations regarding whitespace characters: Whitespace separating words should mark the beginning of a new word, thus occurring in front of the first syllable of a new word. This is also valid for new line characters. A syllable followed by a comma should not be broken apart with a sync (both the syllable and the comma should be before the sync).

An example: The "USLT" passage

> "Strangers in the night" $0A "Exchanging glances"

would be "SYLT" encoded as:

> "Strang" $00 xx xx "ers" $00 xx xx " in" $00 xx xx " the" $00 xx xx " night" $00 xx xx 0A "Ex" $00 xx xx "chang" $00 xx xx "ing" $00 xx xx "glan" $00 xx xx "ces" $00 xx xx

There may be more than one "SYLT" frame in each tag, but only one with the same language and content descriptor.

4.11. Comments

This frame is indended for any kind of full text information that does not fit in any other frame. It consists of a frame header followed by encoding, language and content descriptors and is ended with the actual comment as a text string. Newline characters are allowed in the comment text string. There may be more than one comment frame in each tag, but only one with the same language and content descriptor.

> <Header for 'Comment', ID: "COMM"> Text encoding $xx Language $xx xx xx Short content descrip. <text string according to encoding> $00 (00) The actual text <full text string according to encoding>

4.12. Relative volume adjustment

This is a more subjective function than the previous ones. It allows the user to say how much he wants to increase/decrease the volume on each channel while the file is played. The purpose is to be able to align all files to a reference volume, so that you don't have to change the volume constantly. This frame may also be used to balance adjust the audio. If the volume peak levels are known then this could be described with the 'Peak volume right' and 'Peak volume left' field. If Peakvolume is not known these fields could be left zeroed or, if no other data follows, be completely omitted. There may only be one "RVAD" frame in each tag.

> <Header for 'Relative volume adjustment', ID: "RVAD"> Increment/decrement %00xxxxxx Bits used for volume descr. $xx Relative volume change, right $xx xx (xx ...) Relative volume change, left $xx xx (xx ...) Peak volume right $xx xx (xx ...) Peak volume left $xx xx (xx ...)

In the increment/decrement field bit 0 is used to indicate the right channel and bit 1 is used to indicate the left channel. 1 is increment and 0 is decrement.

The 'bits used for volume description' field is normally $10 (16 bits) for MPEG 2 layer I, II and III [MPEG] and MPEG 2.5. This value may not be $00. The volume is always represented with whole bytes, padded in the beginning (highest bits) when 'bits used for volume description' is not a multiple of eight.

This datablock is then optionally followed by a volume definition for the left and right back channels. If this information is appended to the frame the first two channels will be treated as front channels. In the increment/decrement field bit 2 is used to indicate the right back channel and bit 3 for the left back channel.

> Relative volume change, right back $xx xx (xx ...) Relative volume change, left back $xx xx (xx ...) Peak volume right back $xx xx (xx ...) Peak volume left back $xx xx (xx ...)

If the center channel adjustment is present the following is appended to the existing frame, after the left and right back channels. The center channel is represented by bit 4 in the increase/decrease field.

> Relative volume change, center $xx xx (xx ...) Peak volume center $xx xx (xx ...)

If the bass channel adjustment is present the following is appended to the existing frame, after the center channel. The bass channel is represented by bit 5 in the increase/decrease field.

> Relative volume change, bass $xx xx (xx ...) Peak volume bass $xx xx (xx ...)

4.13. Equalisation

This is another subjective, alignment frame. It allows the user to predefine an equalisation curve within the audio file. There may only be one "EQUA" frame in each tag.

> <Header of 'Equalisation', ID: "EQUA"> Adjustment bits $xx

The 'adjustment bits' field defines the number of bits used for representation of the adjustment. This is normally $10 (16 bits) for MPEG 2 layer I, II and III [MPEG] and MPEG 2.5. This value may not be $00.

This is followed by 2 bytes + ('adjustment bits' rounded up to the nearest byte) for every equalisation band in the following format, giving a frequency range of 0 - 32767Hz:

> Increment/decrement %x (MSB of the Frequency) Frequency (lower 15 bits) Adjustment $xx (xx ...)

The increment/decrement bit is 1 for increment and 0 for decrement. The equalisation bands should be ordered increasingly with reference to frequency. All frequencies don't have to be declared. The equalisation curve in the reading software should be interpolated between the values in this frame. Three equal adjustments for three subsequent frequencies. A frequency should only be described once in the frame.

4.14. Reverb

Yet another subjective one. You may here adjust echoes of different kinds. Reverb left/right is the delay between every bounce in ms. Reverb bounces left/right is the number of bounces that should be made. $FF equals an infinite number of bounces. Feedback is the amount of volume that should be returned to the next echo bounce. $00 is 0%, $FF is 100%. If this value were $7F, there would be 50% volume reduction on the first bounce, 50% of that on the second and so on. Left to left means the sound from the left bounce to be played in the left speaker, while left to right means sound from the left bounce to be played in the right speaker.

'Premix left to right' is the amount of left sound to be mixed in the right before any reverb is applied, where $00 id 0% and $FF is 100%. 'Premix right to left' does the same thing, but right to left. Setting both premix to $FF would result in a mono output (if the reverb is applied symmetric). There may only be one "RVRB" frame in each tag.

> <Header for 'Reverb', ID: "RVRB"> Reverb left (ms) $xx xx Reverb right (ms) $xx xx Reverb bounces, left $xx Reverb bounces, right $xx Reverb feedback, left to left $xx Reverb feedback, left to right $xx Reverb feedback, right to right $xx Reverb feedback, right to left $xx Premix left to right $xx Premix right to left $xx

4.15. Attached picture

This frame contains a picture directly related to the audio file. Image format is the MIME type and subtype [MIME] for the image. In the event that the MIME media type name is omitted, "image/" will be implied. The "image/png" [PNG] or "image/jpeg" [JFIF] picture format should be used when interoperability is wanted. Description is a short description of the picture, represented as a terminated textstring. The description has a maximum length of 64 characters, but may be empty. There may be several pictures

attached to one file, each in their individual "APIC" frame, but only one with the same content descriptor. There may only be one picture with the picture type declared as picture type $01 and $02 respectively. There is the possibility to put only a link to the image file by using the 'MIME type' "–>" and having a complete URL [URL] instead of picture data. The use of linked files should however be used sparingly since there is the risk of separation of files.

<Header for 'Attached picture', ID: "APIC"> Text encoding $xx MIME type <text string> $00 Picture type $xx Description <text string according to encoding> $00 (00) Picture data <binary data>

Picture type: $00 Other $01 32x32 pixels 'file icon' (PNG only) $02 Other file icon $03 Cover (front) $04 Cover (back) $05 Leaflet page $06 Media (e.g. lable side of CD) $07 Lead artist/lead performer/soloist $08 Artist/performer $09 Conductor $0A Band/Orchestra $0B Composer $0C Lyricist/text writer $0D Recording Location $0E During recording $0F During performance $10 Movie/video screen capture $11 A bright coloured fish $12 Illustration $13 Band/artist logotype $14 Publisher/Studio logotype

4.16. General encapsulated object

In this frame any type of file can be encapsulated. After the header, 'Frame size' and 'Encoding' follows 'MIME type' [MIME] represented as as a terminated string encoded with ISO 8859-1 [ISO-8859-1]. The filename is case sensitive and is encoded as 'Encoding'. Then follows a content description as terminated string, encoded as 'Encoding'. The last thing in the frame is the actual object. The first two strings may be omitted, leaving only their terminations. MIME type is always an ISO-8859-1 text string. There may be more than one "GEOB" frame in each tag, but only one with the same content descriptor.

<Header for 'General encapsulated object', ID: "GEOB"> Text encoding $xx MIME type <text string> $00 Filename <text string according to encoding> $00 (00) Content description <text string according to enc?ding> $00 (00) Encapsulated object <binary data>

4.17. Play counter

This is simply a counter of the number of times a file has been played. The value is increased by one every time the file begins to play. There may only be one "PCNT" frame in each tag. When the counter reaches all one's, one byte is inserted in front of the counter thus making the counter eight bits bigger. The counter must be at least 32-bits long to begin with.

<Header for 'Play counter', ID: "PCNT"> Counter $xx xx xx xx (xx ...)

4.18. Popularimeter

The purpose of this frame is to specify how good an audio file is. Many interesting applications could be found to this frame such as a playlist that features better audiofiles more often than others or it could be used to profile a person's taste and find other 'good' files by comparing people's profiles. The frame is very simple. It contains the email address to the user, one rating byte and a four byte play counter, intended to be increased with one for every time the file is played. The email is a terminated string. The rating is 1-255 where 1 is worst and 255 is best. 0 is unknown. If no personal counter is wanted it may be omitted. When the counter reaches all one's, one byte is inserted in front of the counter thus making the counter eight bits bigger in the same away as the play counter ("PCNT"). There may be more than one "POPM" frame in each tag, but only one with the same email address.

<Header for 'Popularimeter', ID: "POPM"> Email to user <text string> $00 Rating $xx Counter $xx xx xx xx (xx ...)

4.19. Recommended buffer size

Sometimes the server from which a audio file is streamed is aware of transmission or coding problems resulting in interruptions in the audio stream. In these cases, the size of the buffer can be recommended by the server using this frame. If the 'embedded info flag' is true (1) then this indicates that an ID3 tag with the maximum size described in 'Buffer size' may occur in the audiostream. In such case the tag should

reside between two MPEG [MPEG] frames, if the audio is MPEG encoded. If the position of the next tag is known, 'offset to next tag' may be used. The offset is calculated from the end of tag in which this frame resides to the first byte of the header in the next. This field may be omitted. Embedded tags are generally not recommended since this could render unpredictable behaviour from present software/hardware.

For applications like streaming audio it might be an idea to embed tags into the audio stream though. If the clients connects to individual connections like HTTP and there is a possibility to begin every transmission with a tag, then this tag should include a 'recommended buffer size' frame. If the client is connected to a arbitrary point in the stream, such as radio or multicast, then the 'recommended buffer size' frame should be included in every tag. Every tag that is picked up after the initial/first tag is to be considered as an update of the previous one. E.g. if there is a "TIT2" frame in the first received tag and one in the second tag, then the first should be 'replaced' with the second.

The 'Buffer size' should be kept to a minimum. There may only be one "RBUF" frame in each tag.

> <Header for 'Recommended buffer size', ID: "RBUF"> Buffer size $xx xx xx Embedded info flag %0000000x Offset to next tag $xx xx xx xx

4.20. Audio encryption

This frame indicates if the actual audio stream is encrypted, and by whom. Since standardisation of such encrypion scheme is beyond this document, all "AENC" frames begin with a terminated string with a URL containing an email address, or a link to a location where an email address can be found, that belongs to the organisation responsible for this specific encrypted audio file. Questions regarding the encrypted audio should be sent to the email address specified. If a $00 is found directly after the 'Frame size' and the audiofile indeed is encrypted, the whole file may be considered useless.

After the 'Owner identifier', a pointer to an unencrypted part of the audio can be specified. The 'Preview start' and 'Preview length' is described in frames. If no part is unencrypted, these fields should be left zeroed. After the 'preview length' field follows optionally a datablock required for decryption of the audio. There may be more than one "AENC" frames in a tag, but only one with the same 'Owner identifier'.

> <Header for 'Audio encryption', ID: "AENC"> Owner identifier <text string> $00 Preview start $xx xx Preview length $xx xx Encryption info <binary data>

4.21. Linked information

To keep space waste as low as possible this frame may be used to link information from another ID3v2 tag that might reside in another audio file or alone in a binary file. It is recommended that this method is only used when the files are stored on a CD-ROM or other circumstances when the risk of file seperation is low. The frame contains a frame identifier, which is the frame that should be linked into this tag, a URL [URL] field, where a reference to the file where the frame is given, and additional ID data, if needed. Data should be retrieved from the first tag found in the file to which this link points. There may be more than one "LINK" frame in a tag, but only one with the same contents. A linked frame is to be considered as part of the tag and has the same restrictions as if it was a physical part of the tag (i.e. only one "RVRB" frame allowed, whether it's linked or not).

> <Header for 'Linked information', ID: "LINK"> Frame identifier $xx xx xx URL <text string> $00 ID and additional data <text string(s)>

Frames that may be linked and need no additional data are "IPLS", "MCID", "ETCO", "MLLT", "SYTC", "RVAD", "EQUA", "RVRB", "RBUF", the text information frames and the URL link frames.

The "TXXX", "APIC", "GEOB" and "AENC" frames may be linked with the content descriptor as additional ID data.

The "COMM", "SYLT" and "USLT" frames may be linked with three bytes of language descriptor directly followed by a content descriptor as additional ID data.

4.22. Position synchronisation frame

This frame delivers information to the listener of how far into the audio stream he picked up; in effect, it states the time offset of the first frame in the stream. The frame layout is:

<Head for 'Position synchronisation', ID: "POSS"> Time stamp format $xx Position $xx (xx ...)

Where time stamp format is:

$01 Absolute time, 32 bit sized, using MPEG frames as unit $02 Absolute time, 32 bit sized, using milliseconds as unit

and position is where in the audio the listener starts to receive, i.e. the beginning of the next frame. If this frame is used in the beginning of a file the value is always 0. There may only be one "POSS" frame in each tag.

4.23. Terms of use frame

This frame contains a brief description of the terms of use and ownership of the file. More detailed information concerning the legal terms might be available through the "WCOP" frame. Newlines are allowed in the text. There may only be one "USER" frame in a tag.

<Header for 'Terms of use frame', ID: "USER"> Text encoding $xx Language $xx xx xx The actual text <text string according to encoding>

4.24. Ownership frame

The ownership frame might be used as a reminder of a made transaction or, if signed, as proof. Note that the "USER" and "TOWN" frames are good to use in conjunction with this one. The frame begins, after the frame ID, size and encoding fields, with a 'price payed' field. The first three characters of this field contains the currency used for the transaction, encoded according to ISO 4217 [ISO-4217] alphabetic currency code. Concatenated to this is the actual price payed, as a numerical string using "." as the decimal separator. Next is an 8 character date string (YYYYMMDD) followed by a string with the name of the seller as the last field in the frame. There may only be one "OWNE" frame in a tag.

<Header for 'Ownership frame', ID: "OWNE"> Text encoding $xx Price payed <text string> $00 Date of purch. <text string> Seller <text string according to encoding>

4.25. Commercial frame

This frame enables several competing offers in the same tag by bundling all needed information. That makes this frame rather complex but it's an easier solution than if one tries to achieve the same result with several frames. The frame begins, after the frame ID, size and encoding fields, with a price string field. A price is constructed by one three character currency code, encoded according to ISO 4217 [ISO-4217] alphabetic currency code, followed by a numerical value where "." is used as decimal seperator. In the price string several prices may be concatenated, seperated by a "/" character, but there may only be one currency of each type.

The price string is followed by an 8 character date string in the format YYYYMMDD, describing for how long the price is valid. After that is a contact URL, with which the user can contact the seller, followed by a one byte 'received as' field. It describes how the audio is delivered when bought according to the following list:

$00 Other $01 Standard CD album with other songs $02 Compressed audio on CD $03 File over the Internet $04 Stream over the Internet $05 As note sheets $06 As note sheets in a book with other sheets $07 Music on other media $08 Non-musical merchandise

Next follows a terminated string with the name of the seller followed by a terminated string with a short description of the product. The last thing is the ability to include a company logotype. The first of them is the 'Picture MIME type' field containing information about which picture format is used. In the event that the MIME media type name is omitted, "image/" will be implied. Currently only "image/png" and

"image/jpeg" are allowed. This format string is followed by the binary picture data. This two last fields may be omitted if no picture is to attach.

<Header for 'Commercial frame', ID: "COMR"> Text encoding $xx Price string <text string> $00 Valid until <text string> Contact URL <text string> $00 Received as $xx Name of seller <text string according to encoding> $00 (00) Description <text string according to encoding> $00 (00) Picture MIME type <string> $00 Seller logo <binary data>

4.26. Encryption method registration

To identify with which method a frame has been encrypted the encryption method must be registered in the tag with this frame. The 'Owner identifier' is a null-terminated string with a URL [URL] containing an email address, or a link to a location where an email address can be found, that belongs to the organisation responsible for this specific encryption method. Questions regarding the encryption method should be sent to the indicated email address. The 'Method symbol' contains a value that is associated with this method throughout the whole tag. Values below $80 are reserved. The 'Method symbol' may optionally be followed by encryption specific data. There may be several "ENCR" frames in a tag but only one containing the same symbol and only one containing the same owner identifier. The method must be used somewhere in the tag. See section 3.3.1, flag j for more information.

<Header for 'Encryption method registration', ID: "ENCR"> Owner identifier <text string> $00 Method symbol $xx Encryption data <binary data>

4.27. Group identification registration

This frame enables grouping of otherwise unrelated frames. This can be used when some frames are to be signed. To identify which frames belongs to a set of frames a group identifier must be registered in the tag with this frame. The 'Owner identifier' is a null-terminated string with a URL [URL] containing an email address, or a link to a location where an email address can be found, that belongs to the organisation responsible for this grouping. Questions regarding the grouping should be sent to the indicated email address. The 'Group symbol' contains a value that associates the frame with this group throughout the whole tag. Values below $80 are reserved. The 'Group symbol' may optionally be followed by some group specific data, e.g. a digital signature. There may be several "GRID" frames in a tag but only one containing the same symbol and only one containing the same owner identifier. The group symbol must be used somewhere in the tag. See section 3.3.1, flag j for more information.

<Header for 'Group ID registration', ID: "GRID"> Owner identifier <text string> $00 Group symbol $xx

Group dependent data <binary data>

4.28. Private frame

This frame is used to contain information from a software producer that its program uses and does not fit into the other frames. The frame consists of an 'Owner identifier' string and the binary data. The 'Owner identifier' is a null-terminated string with a URL [URL] containing an email address, or a link to a location where an email address can be found, that belongs to the organisation responsible for the frame. Questions regarding the frame should be sent to the indicated email address. The tag may contain more than one "PRIV" frame but only with different contents. It is recommended to keep the number of "PRIV" frames as low as possible.

<Header for 'Private frame', ID: "PRIV"> Owner identifier <text string> $00

The private data <binary data>

5. The 'unsynchronisation scheme'

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

Whenever a false synchronisation is found within the tag, one zeroed byte is inserted after the first false synchronisation byte. The format of a correct sync 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 won't 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 first bit in 'ID3 flags' should be set. This bit should only be set if the tag contains a, now corrected, false synchronisation. The bit should only be clear if the tag does not contain any false synchronisations.

Do bear in mind, that if a compression scheme is used by the encoder, the unsynchronisation scheme should be applied *afterwards*. When decoding a compressed, 'unsynchronised' file, the 'unsynchronisation scheme' should be parsed first, decompression afterwards.

If the last byte in the tag is $FF, and there is a need to eliminate false synchronisations in the tag, at least one byte of padding should be added.

6. Copyright

7. References

[CDDB] Compact Disc Data Base

http://www.cddb.com

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

http://www.id3lib.org/id3/id3v2-00.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-4217] ISO 4217:1995. Codes for the representation of currencies and funds. Technical committee / subcommittee: TC 68

[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

[ISRC] ISO 3901:1986 International Standard Recording Code (ISRC). Technical committee / subcommittee: TC 46 / SC 9

[JFIF] JPEG File Interchange Format, version 1.02

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

[MIME] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, November 1996.

ftp://ftp.isi.edu/in-notes/rfc2045.txt">ftp://ftp.isi.edu/in-notes/rfc2045.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

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

[UNICODE] ISO/IEC 10646-1:1993. Universal Multiple-Octet Coded Character Set (UCS), Part 1: Architecture and Basic Multilingual Plane. Technical committee / subcommittee: JTC 1 / SC 2

http://www.unicode.org/

[URL] T. Berners-Lee, L. Masinter & M. McCahill, "Uniform Resource Locators (URL).", RFC 1738, December 1994.

ftp://ftp.isi.edu/in-notes/rfc1738.txt

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

ftp://ftp.isi.edu/in-notes/rfc1950.txt

8. Appendix

1. Appendix A - Genre List from ID3v1

The following genres is defined in ID3v1

0.Blues 1.Classic Rock 2.Country 3.Dance 4.Disco 5.Funk 6.Grunge 7.Hip-Hop 8.Jazz 9.Metal

10.New Age 11.Oldies 12.Other 13.Pop 14.R&B 15.Rap 16.Reggae 17.Rock 18.Techno 19.Industrial 20.Alternative 21.Ska 22.Death Metal 23.Pranks 24.Soundtrack 25.Euro-Techno 26.Ambient 27.Trip-Hop 28.Vocal 29.Jazz+Funk 30.Fusion 31.Trance 32.Classical 33.Instrumental 34.Acid 35.House 36.Game 37.Sound Clip 38.Gospel 39.Noise 40.AlternRock 41.Bass 42.Soul 43.Punk 44.Space 45.Meditative 46.Instrumental Pop 47.Instrumental Rock 48.Ethnic 49.Gothic 50.Darkwave 51.Techno-Industrial 52.Electronic 53.Pop-Folk 54.Eurodance 55.Dream 56.Southern Rock 57.Comedy 58.Cult 59.Gangsta 60.Top 40 61.Christian Rap 62.Pop/Funk 63.Jungle 64.Native American 65.Cabaret 66.New Wave 67.Psychadelic 68.Rave 69.Showtunes 70.Trailer 71.Lo-Fi 72.Tribal 73.Acid Punk 74.Acid Jazz 75.Polka 76.Retro 77.Musical 78.Rock & Roll 79.Hard Rock

The following genres are Winamp extensions

80.Folk 81.Folk-Rock 82.National Folk 83.Swing 84.Fast Fusion 85.Bebob 86.Latin 87.Revival 88.Celtic 89.Bluegrass 90.Avantgarde 91.Gothic Rock 92.Progressive Rock 93.Psychedelic Rock 94.Symphonic Rock 95.Slow Rock 96.Big Band 97.Chorus 98.Easy Listening 99.Acoustic

100.Humour 101.Speech 102.Chanson 103.Opera 104.Chamber Music 105.Sonata 106.Symphony 107.Booty Bass 108.Primus 109.Porn Groove 110.Satire 111.Slow Jam 112.Club 113.Tango 114.Samba 115.Folklore 116.Ballad 117.Power Ballad 118.Rhythmic Soul 119.Freestyle 120.Duet 121.Punk Rock 122.Drum Solo 123.Acapella 124.Euro-House 125.Dance Hall

9. Author's Address

Written by

Martin Nilsson Rydsv?gen 246 C. 30 S-584 34 Link?ping Sweden

Email: nilsson@id3.org

Edited by

Dirk Mahoney 57 Pechey Street Chermside Q Australia 4032

Email: dirk@id3.org

Johan Sundstr?m Als?ttersgatan 5 A. 34 S-584 35 Link?ping Sweden

Email: johan@id3.org

1.3 ID3 tag version 2.4.0 - Main Structure

1.3.1 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.

1.3.2 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.

1.3.3 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].

1.3.4 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:

Header (10 bytes)
Extended Header (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.

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.

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.
```

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.

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
```

1.3.5 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.

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.

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.

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.
```

1.3.6 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.

1.3.7 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.

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.

1.3.8 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).
```

1.3.9 Copyright

1.3.10 References

ID3v2 Martin Nilsson, ID3v2 informal standard.

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

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

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

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

URL T. Berners-Lee, L. Masinter & M. McCahill, Uniform Resource Locators (URL), RFC 1738, December 1994.

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

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

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

1.3.11 Author's Address

Written by

Martin Nilsson
Rydsvägen 246 C. 30
SE-584 34 Linköping
Sweden

Email: nilsson at id3.org

1.4 ID3 tag version 2.4.0 - Native Frames

1.4.1 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.

1.4.2 Abstract

This document describes the frames natively supported by ID3v2.4.0, which is a revised version of the ID3v2 informal standard [*ID3v2.3.0*] version 2.3.0. The ID3v2 offers a flexible way of storing audio meta information within 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.

1.4.3 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 the ID3v2 main structure document [*ID3v2-strct*]. 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*].

1.4.4 Default flags

The default settings for the frames described in this document can be divided into the following classes. The flags may be set differently if found more suitable by the software.

1. Discarded if tag is altered, discarded if file is altered.

   None.

2. Discarded if tag is altered, preserved if file is altered.

   None.

3. Preserved if tag is altered, discarded if file is altered.

   ASPI, *AENC*, *ETCO*, *EQU2*, *MLLT*, *POSS*, *SEEK*, *SYLT*, *SYTC*, *RVA2*, *TENC*, *TLEN*

4. Preserved if tag is altered, preserved if file is altered.

   The rest of the frames.

1.4.5 Declared ID3v2 frames

The following frames are declared in this draft.

- *AENC* Audio encryption
- *APIC* Attached picture

- *ASPI* Audio seek point index
- *COMM* Comments
- *COMR* Commercial frame
- *ENCR* Encryption method registration
- *EQU2* Equalisation (2)
- *ETCO* Event timing codes
- *GEOB* General encapsulated object
- *GRID* Group identification registration
- *LINK* Linked information
- *MCDI* Music CD identifier
- *MLLT* MPEG location lookup table
- *OWNE* Ownership frame
- *PRIV* Private frame
- *PCNT* Play counter
- *POPM* Popularimeter
- *POSS* Position synchronisation frame
- *RBUF* Recommended buffer size
- *RVA2* Relative volume adjustment (2)
- *RVRB* Reverb
- *SEEK* Seek frame
- *SIGN* Signature frame
- *SYLT* Synchronised lyric/text
- *SYTC* Synchronised tempo codes
- *TALB* Album/Movie/Show title
- *TBPM* BPM (beats per minute)
- *TCOM* Composer
- *TCON* Content type
- *TCOP* Copyright message
- *TDEN* Encoding time
- *TDLY* Playlist delay
- *TDOR* Original release time
- *TDRC* Recording time
- *TDRL* Release time
- *TDTG* Tagging time
- *TENC* Encoded by
- *TEXT* Lyricist/Text writer

- *TFLT* File type
- *TIPL* Involved people list
- *TIT1* Content group description
- *TIT2* Title/songname/content description
- *TIT3* Subtitle/Description refinement
- *TKEY* Initial key
- *TLAN* Language(s)
- *TLEN* Length
- *TMCL* Musician credits list
- *TMED* Media type
- *TMOO* Mood
- *TOAL* Original album/movie/show title
- *TOFN* Original filename
- *TOLY* Original lyricist(s)/text writer(s)
- *TOPE* Original artist(s)/performer(s)
- *TOWN* File owner/licensee
- *TPE1* Lead performer(s)/Soloist(s)
- *TPE2* Band/orchestra/accompaniment
- *TPE3* Conductor/performer refinement
- *TPE4* Interpreted, remixed, or otherwise modified by
- *TPOS* Part of a set
- *TPRO* Produced notice
- *TPUB* Publisher
- *TRCK* Track number/Position in set
- *TRSN* Internet radio station name
- *TRSO* Internet radio station owner
- *TSOA* Album sort order
- *TSOP* Performer sort order
- *TSOT* Title sort order
- *TSRC* ISRC (international standard recording code)
- *TSSE* Software/Hardware and settings used for encoding
- *TSST* Set subtitle
- *TXXX* User defined text information frame
- *UFID* Unique file identifier
- *USER* Terms of use
- *USLT* Unsynchronised lyric/text transcription

- *WCOM* Commercial information
- *WCOP* Copyright/Legal information
- *WOAF* Official audio file webpage
- *WOAR* Official artist/performer webpage
- *WOAS* Official audio source webpage
- *WORS* Official Internet radio station homepage
- *WPAY* Payment
- *WPUB* Publishers official webpage
- *WXXX* User defined URL link frame

Unique file identifier

This frame's purpose is to be able to identify the audio file in a database, that may provide more information relevant to the content. Since standardisation of such a database is beyond this document, all *UFID* frames begin with an 'owner identifier' field. It is a null- terminated string with a URL [*URL*] containing an email address, or a link to a location where an email address can be found, that belongs to the organisation responsible for this specific database implementation. Questions regarding the database should be sent to the indicated email address. The URL should not be used for the actual database queries. The string "http://www.id3.org/dummy/ufid.html" should be used for tests. The 'Owner identifier' must be non-empty (more than just a termination). The 'Owner identifier' is then followed by the actual identifier, which may be up to 64 bytes. There may be more than one "UFID" frame in a tag, but only one with the same 'Owner identifier'.

```
<Header for 'Unique file identifier', ID: "UFID">
Owner identifier        <text string> $00
Identifier              <up to 64 bytes binary data>
```

Text information frames

The text information frames are often the most important frames, containing information like artist, album and more. There may only be one text information frame of its kind in an tag. All text information frames supports multiple strings, stored as a null separated list, where null is reperesented by the termination code for the charater encoding. All text frame identifiers begin with "T". Only text frame identifiers begin with "T", with the exception of the "TXXX" frame. All the text information frames have the following format:

```
<Header for 'Text information frame', ID: "T000" - "TZZZ",
excluding "TXXX" described in 4.2.6.>
Text encoding               $xx
Information                 <text string(s) according to encoding>
```

Identification frames

TIT1 The 'Content group description' frame is used if the sound belongs to a larger category of sounds/music. For example, classical music is often sorted in different musical sections (e.g. "Piano Concerto", "Weather - Hurricane").

TIT2 The 'Title/Songname/Content description' frame is the actual name of the piece (e.g. "Adagio", "Hurricane Donna").

TIT3 The 'Subtitle/Description refinement' frame is used for information directly related to the contents title (e.g. "Op. 16" or "Performed live at Wembley").

TALB The 'Album/Movie/Show title' frame is intended for the title of the recording (or source of sound) from which the audio in the file is taken.

TOAL The 'Original album/movie/show title' frame is intended for the title of the original recording (or source of sound), if for example the music in the file should be a cover of a previously released song.

TRCK The 'Track number/Position in set' frame is a numeric string containing the order number of the audio-file on its original recording. This MAY be extended with a "/" character and a numeric string containing the total number of tracks/elements on the original recording. E.g. "4/9".

TPOS The 'Part of a set' frame is a numeric string that describes which part of a set the audio came from. This frame is used if the source described in the "TALB" frame is divided into several mediums, e.g. a double CD. The value MAY be extended with a "/" character and a numeric string containing the total number of parts in the set. E.g. "1/2".

TSST The 'Set subtitle' frame is intended for the subtitle of the part of a set this track belongs to.

TSRC The 'ISRC' frame should contain the International Standard Recording Code [*ISRC*] (12 characters).

Involved persons frames

TPE1 The 'Lead artist/Lead performer/Soloist/Performing group' is used for the main artist.

TPE2 The 'Band/Orchestra/Accompaniment' frame is used for additional information about the performers in the recording.

TPE3 The 'Conductor' frame is used for the name of the conductor.

TPE4 The 'Interpreted, remixed, or otherwise modified by' frame contains more information about the people behind a remix and similar interpretations of another existing piece.

TOPE The 'Original artist/performer' frame is intended for the performer of the original recording, if for example the music in the file should be a cover of a previously released song.

TEXT The 'Lyricist/Text writer' frame is intended for the writer of the text or lyrics in the recording.

TOLY The 'Original lyricist/text writer' frame is intended for the text writer of the original recording, if for example the music in the file should be a cover of a previously released song.

TCOM The 'Composer' frame is intended for the name of the composer.

TMCL The 'Musician credits list' is intended as a mapping between instruments and the musician that played it. Every odd field is an instrument and every even is an artist or a comma delimited list of artists.

TIPL The 'Involved people list' is very similar to the musician credits list, but maps between functions, like producer, and names.

TENC The 'Encoded by' frame contains the name of the person or organisation that encoded the audio file. This field may contain a copyright message, if the audio file also is copyrighted by the encoder.

Derived and subjective properties frames

TBPM The 'BPM' frame contains the number of beats per minute in the main part of the audio. The BPM is an integer and represented as a numerical string.

TLEN The 'Length' frame contains the length of the audio file in milliseconds, represented as a numeric string.

TKEY The 'Initial key' frame contains the musical key in which the sound starts. It is represented as a string with a maximum length of three characters. The ground keys are represented with "A","B","C","D","E", "F" and "G" and halfkeys represented with "b" and "#". Minor is represented as "m", e.g. "Dbm" $00. Off key is represented with an "o" only.

TLAN The 'Language' frame should contain the languages of the text or lyrics spoken or sung in the audio. The language is represented with three characters according to ISO-639-2 [*ISO-639-2*]. If more than one language is used in the text their language codes should follow according to the amount of their usage, e.g. "eng" $00 "sve" $00.

TCON The 'Content type', which ID3v1 was stored as a one byte numeric value only, is now a string. You may use one or several of the ID3v1 types as numerical strings, or, since the category list would be impossible to maintain with accurate and up to date categories, define your own. Example: "21" $00 "Eurodisco" $00

You may also use any of the following keywords:

RX Remix
CR Cover

TFLT

The 'File type' frame indicates which type of audio this tag defines. The following types and refinements are defined:

MIME MIME type follows
MPG MPEG Audio

/1	MPEG 1/2 layer I
/2	MPEG 1/2 layer II
/3	MPEG 1/2 layer III
/2.5	MPEG 2.5
/AAC	Advanced audio compression

VQF Transform-domain Weighted Interleave Vector Quantisation
PCM Pulse Code Modulated audio

but other types may be used, but not for these types though. This is used in a similar way to the predefined types in the "*TMED*" frame, but without parentheses. If this frame is not present audio type is assumed to be "MPG".

TMED The 'Media type' frame describes from which media the sound originated. This may be a text string or a reference to the predefined media types found in the list below. Example: "VID/PAL/VHS" $00.

DIG Other digital media

/A	Analogue transfer from media

ANA Other analogue media

/WAC	Wax cylinder
/8CA	8-track tape cassette

CD CD

/A	Analogue transfer from media
/DD	DDD
/AD	ADD
/AA	AAD

LD Laserdisc

TT Turntable records

/33	33.33 rpm
/45	45 rpm
/71	71.29 rpm
/76	76.59 rpm
/78	78.26 rpm
/80	80 rpm

MD MiniDisc

/A	Analogue transfer from media

DAT DAT

/A	Analogue transfer from media
/1	standard, 48 kHz/16 bits, linear
/2	mode 2, 32 kHz/16 bits, linear
/3	mode 3, 32 kHz/12 bits, non-linear, low speed
/4	mode 4, 32 kHz/12 bits, 4 channels
/5	mode 5, 44.1 kHz/16 bits, linear
/6	mode 6, 44.1 kHz/16 bits, 'wide track' play

DCC DCC

/A	Analogue transfer from media

DVD DVD

/A	Analogue transfer from media

TV Television

/PAL	PAL
/NTSC	NTSC
/SECAM	SECAM

VID Video

/PAL	PAL
/NTSC	NTSC
/SECAM	SECAM
/VHS	VHS
/SVHS	S-VHS
/BETA	BETAMAX

RAD Radio

/FM	FM
/AM	AM

/LW	LW
/MW	MW

TEL Telephone

/I	ISDN

MC MC (normal cassette)

/4	4.75 cm/s (normal speed for a two sided cassette)
/9	9.5 cm/s
/I	Type I cassette (ferric/normal)
/II	Type II cassette (chrome)
/III	Type III cassette (ferric chrome)
/IV	Type IV cassette (metal)

REE Reel

/9	9.5 cm/s
/19	19 cm/s
/38	38 cm/s
/76	76 cm/s
/I	Type I cassette (ferric/normal)
/II	Type II cassette (chrome)
/III	Type III cassette (ferric chrome)
/IV	Type IV cassette (metal)

TMOO The 'Mood' frame is intended to reflect the mood of the audio with a few keywords, e.g. "Romantic" or "Sad".

Rights and license frames

TCOP The 'Copyright message' frame, in which the string must begin with a year and a space character (making five characters), is intended for the copyright holder of the original sound, not the audio file itself. The absence of this frame means only that the copyright information is unavailable or has been removed, and must not be interpreted to mean that the audio is public domain. Every time this field is displayed the field must be preceded with "Copyright " (C) " ", where (C) is one character showing a C in a circle.

TPRO The 'Produced notice' frame, in which the string must begin with a year and a space character (making five characters), is intended for the production copyright holder of the original sound, not the audio file itself. The absence of this frame means only that the production copyright information is unavailable or has been removed, and must not be interpreted to mean that the audio is public domain. Every time this field is displayed the field must be preceded with "Produced " (P) " ", where (P) is one character showing a P in a circle.

TPUB The 'Publisher' frame simply contains the name of the label or publisher.

TOWN The 'File owner/licensee' frame contains the name of the owner or licensee of the file and it's contents.

TRSN The 'Internet radio station name' frame contains the name of the internet radio station from which the audio is streamed.

TRSO The 'Internet radio station owner' frame contains the name of the owner of the internet radio station from which the audio is streamed.

Other text frames

TOFN The 'Original filename' frame contains the preferred filename for the file, since some media doesn't allow the desired length of the filename. The filename is case sensitive and includes its suffix.

TDLY The 'Playlist delay' defines the numbers of milliseconds of silence that should be inserted before this audio. The value zero indicates that this is a part of a multifile audio track that should be played continuously.

TDEN The 'Encoding time' frame contains a timestamp describing when the audio was encoded. Timestamp format is described in the ID3v2 structure document [*ID3v2-strct*].

TDOR The 'Original release time' frame contains a timestamp describing when the original recording of the audio was released. Timestamp format is described in the ID3v2 structure document [*ID3v2-strct*].

TDRC The 'Recording time' frame contains a timestamp describing when the audio was recorded. Timestamp format is described in the ID3v2 structure document [*ID3v2-strct*].

TDRL The 'Release time' frame contains a timestamp describing when the audio was first released. Timestamp format is described in the ID3v2 structure document [*ID3v2-strct*].

TDTG The 'Tagging time' frame contains a timestamp describing then the audio was tagged. Timestamp format is described in the ID3v2 structure document [*ID3v2-strct*].

TSSE The 'Software/Hardware and settings used for encoding' frame includes the used audio encoder and its settings when the file was encoded. Hardware refers to hardware encoders, not the computer on which a program was run.

TSOA The 'Album sort order' frame defines a string which should be used instead of the album name (*TALB*) for sorting purposes. E.g. an album named "A Soundtrack" might preferably be sorted as "Soundtrack".

TSOP The 'Performer sort order' frame defines a string which should be used instead of the performer (*TPE2*) for sorting purposes.

TSOT The 'Title sort order' frame defines a string which should be used instead of the title (*TIT2*) for sorting purposes.

User defined text information frame

This frame is intended for one-string text information concerning the audio file in a similar way to the other "T"-frames. The frame body consists of a description of the string, represented as a terminated string, followed by the actual string. There may be more than one "TXXX" frame in each tag, but only one with the same description.

```
<Header for 'User defined text information frame', ID: "TXXX">
Text encoding     $xx
Description       <text string according to encoding> $00 (00)
Value             <text string according to encoding>
```

URL link frames

With these frames dynamic data such as webpages with touring information, price information or plain ordinary news can be added to the tag. There may only be one URL [URL] link frame of its kind in an tag, except when stated otherwise in the frame description. If the text string is followed by a string termination, all the following information should be ignored and not be displayed. All URL link frame identifiers begins with "W". Only URL link frame identifiers begins with "W", except for "WXXX". All URL link frames have the following format:

```
<Header for 'URL link frame', ID: "W000" - "WZZZ", excluding "WXXX"
described in 4.3.2.>
URL              <text string>
```

URL link frames - details

WCOM The 'Commercial information' frame is a URL pointing at a webpage with information such as where the album can be bought. There may be more than one "WCOM" frame in a tag, but not with the same content.

WCOP The 'Copyright/Legal information' frame is a URL pointing at a webpage where the terms of use and ownership of the file is described.

WOAF The 'Official audio file webpage' frame is a URL pointing at a file specific webpage.

WOAR The 'Official artist/performer webpage' frame is a URL pointing at the artists official webpage. There may be more than one "WOAR" frame in a tag if the audio contains more than one performer, but not with the same content.

WOAS The 'Official audio source webpage' frame is a URL pointing at the official webpage for the source of the audio file, e.g. a movie.

WORS The 'Official Internet radio station homepage' contains a URL pointing at the homepage of the internet radio station.

WPAY The 'Payment' frame is a URL pointing at a webpage that will handle the process of paying for this file.

WPUB The 'Publishers official webpage' frame is a URL pointing at the official webpage for the publisher.

User defined URL link frame

This frame is intended for URL [URL] links concerning the audio file in a similar way to the other "W"-frames. The frame body consists of a description of the string, represented as a terminated string, followed by the actual URL. The URL is always encoded with ISO-8859-1 [ISO-8859-1]. There may be more than one "WXXX" frame in each tag, but only one with the same description.

```
<Header for 'User defined URL link frame', ID: "WXXX">
Text encoding     $xx
Description       <text string according to encoding> $00 (00)
URL               <text string>
```

Music CD identifier

This frame is intended for music that comes from a CD, so that the CD can be identified in databases such as the CDDB [CDDB]. The frame consists of a binary dump of the Table Of Contents, TOC, from the CD, which is a header of 4 bytes and then 8 bytes/track on the CD plus 8 bytes for the 'lead out', making a maximum of 804 bytes. The offset to the beginning of every track on the CD should be described with a four bytes absolute CD-frame address per track, and not with absolute time. When this frame is used the presence of a valid "TRCK" frame is REQUIRED, even if the CD's only got one track. It is recommended that this frame is always added to tags originating from CDs. There may only be one "MCDI" frame in each tag.

```
<Header for 'Music CD identifier', ID: "MCDI">
CD TOC                <binary data>
```

Event timing codes

This frame allows synchronisation with key events in the audio. The header is:

```
<Header for 'Event timing codes', ID: "ETCO">
Time stamp format    $xx
```

Where time stamp format is:

```
$01  Absolute time, 32 bit sized, using MPEG [MPEG] frames as unit
$02  Absolute time, 32 bit sized, using milliseconds as unit
```

Absolute time means that every stamp contains the time from the beginning of the file.

Followed by a list of key events in the following format:

```
Type of event   $xx
Time stamp      $xx (xx ...)
```

The 'Time stamp' is set to zero if directly at the beginning of the sound or after the previous event. All events MUST be sorted in chronological order. The type of event is as follows:

```
$00  padding (has no meaning)
$01  end of initial silence
$02  intro start
$03  main part start
$04  outro start
$05  outro end
$06  verse start
$07  refrain start
$08  interlude start
$09  theme start
$0A  variation start
$0B  key change
$0C  time change
$0D  momentary unwanted noise (Snap, Crackle & Pop)
$0E  sustained noise
$0F  sustained noise end
$10  intro end
$11  main part end
$12  verse end
$13  refrain end
$14  theme end
$15  profanity
$16  profanity end

$17-$DF  reserved for future use

$E0-$EF  not predefined synch 0-F

$F0-$FC  reserved for future use

$FD  audio end (start of silence)
$FE  audio file ends
$FF  one more byte of events follows (all the following bytes with
     the value $FF have the same function)
```

Terminating the start events such as "intro start" is OPTIONAL. The 'Not predefined synch's ($E0-EF) are for user events. You might want to synchronise your music to something, like setting off an explosion on-stage, activating a screensaver etc.

There may only be one "ETCO" frame in each tag.

MPEG location lookup table

To increase performance and accuracy of jumps within a MPEG [MPEG] audio file, frames with time codes in different locations in the file might be useful. This ID3v2 frame includes references that the software can use to calculate positions in the file. After the frame header follows a descriptor of how much the 'frame counter' should be increased for every reference. If this value is two then the first reference points out the second frame, the 2nd reference the 4th frame, the 3rd reference the 6th frame etc. In a similar way the 'bytes between reference' and 'milliseconds between reference' points out bytes and milliseconds respectively.

Each reference consists of two parts; a certain number of bits, as defined in 'bits for bytes deviation', that describes the difference between what is said in 'bytes between reference' and the reality and a certain number of bits, as defined in 'bits for milliseconds deviation', that describes the difference between what is said in 'milliseconds between reference' and the reality. The number of bits in every reference, i.e. 'bits for bytes deviation'+'bits for milliseconds deviation', must be a multiple of four. There may only be one "MLLT" frame in each tag.

```
<Header for 'Location lookup table', ID: "MLLT">
MPEG frames between reference  $xx xx
Bytes between reference        $xx xx xx
Milliseconds between reference $xx xx xx
Bits for bytes deviation       $xx
Bits for milliseconds dev.     $xx
```

Then for every reference the following data is included;

```
Deviation in bytes         %xxx....
Deviation in milliseconds  %xxx....
```

Synchronised tempo codes

For a more accurate description of the tempo of a musical piece, this frame might be used. After the header follows one byte describing which time stamp format should be used. Then follows one or more tempo codes. Each tempo code consists of one tempo part and one time part. The tempo is in BPM described with one or two bytes. If the first byte has the value $FF, one more byte follows, which is added to the first giving a range from 2 - 510 BPM, since $00 and $01 is reserved. $00 is used to describe a beat-free time period, which is not the same as a music-free time period. $01 is used to indicate one single beat-stroke followed by a beat-free period.

The tempo descriptor is followed by a time stamp. Every time the tempo in the music changes, a tempo descriptor may indicate this for the player. All tempo descriptors MUST be sorted in chronological order. The first beat-stroke in a time-period is at the same time as the beat description occurs. There may only be one "SYTC" frame in each tag.

```
<Header for 'Synchronised tempo codes', ID: "SYTC">
Time stamp format   $xx
Tempo data          <binary data>
```

Where time stamp format is:

```
$01  Absolute time, 32 bit sized, using MPEG [MPEG] frames as unit
$02  Absolute time, 32 bit sized, using milliseconds as unit
```

Absolute time means that every stamp contains the time from the beginning of the file.

Unsynchronised lyrics/text transcription

This frame contains the lyrics of the song or a text transcription of other vocal activities. The head includes an encoding descriptor and a content descriptor. The body consists of the actual text. The 'Content descriptor' is a terminated string. If no descriptor is entered, 'Content descriptor' is $00 (00) only. Newline characters are allowed in the text. There may

be more than one 'Unsynchronised lyrics/text transcription' frame in each tag, but only one with the same language and content descriptor.

```
<Header for 'Unsynchronised lyrics/text transcription', ID: "USLT">
Text encoding        $xx
Language             $xx xx xx
Content descriptor   <text string according to encoding> $00 (00)
Lyrics/text          <full text string according to encoding>
```

Synchronised lyrics/text

This is another way of incorporating the words, said or sung lyrics, in the audio file as text, this time, however, in sync with the audio. It might also be used to describing events e.g. occurring on a stage or on the screen in sync with the audio. The header includes a content descriptor, represented with as terminated text string. If no descriptor is entered, 'Content descriptor' is $00 (00) only.

```
<Header for 'Synchronised lyrics/text', ID: "SYLT">
Text encoding        $xx
Language             $xx xx xx
Time stamp format    $xx
Content type         $xx
Content descriptor   <text string according to encoding> $00 (00)
```

Content type:

```
$00 is other
$01 is lyrics
$02 is text transcription
$03 is movement/part name (e.g. "Adagio")
$04 is events (e.g. "Don Quijote enters the stage")
$05 is chord (e.g. "Bb F Fsus")
$06 is trivia/'pop up' information
$07 is URLs to webpages
$08 is URLs to images
```

Time stamp format:

```
$01  Absolute time, 32 bit sized, using MPEG [MPEG] frames as unit
$02  Absolute time, 32 bit sized, using milliseconds as unit
```

Absolute time means that every stamp contains the time from the beginning of the file.

The text that follows the frame header differs from that of the unsynchronised lyrics/text transcription in one major way. Each syllable (or whatever size of text is considered to be convenient by the encoder) is a null terminated string followed by a time stamp denoting where in the sound file it belongs. Each sync thus has the following structure:

```
Terminated text to be synced (typically a syllable)
Sync identifier (terminator to above string)   $00 (00)
Time stamp                                     $xx (xx ...)
```

The 'time stamp' is set to zero or the whole sync is omitted if located directly at the beginning of the sound. All time stamps should be sorted in chronological order. The sync can be considered as a validator of the subsequent string.

Newline characters are allowed in all "SYLT" frames and MUST be used after every entry (name, event etc.) in a frame with the content type $03 - $04.

A few considerations regarding whitespace characters: Whitespace separating words should mark the beginning of a new word, thus occurring in front of the first syllable of a new word. This is also valid for new line characters. A

syllable followed by a comma should not be broken apart with a sync (both the syllable and the comma should be before the sync).

An example: The "USLT" passage

```
"Strangers in the night" $0A "Exchanging glances"
```

would be "SYLT" encoded as:

```
"Strang" $00 xx xx "ers" $00 xx xx " in" $00 xx xx " the" $00 xx xx
" night" $00 xx xx 0A "Ex" $00 xx xx "chang" $00 xx xx "ing" $00 xx
xx "glan" $00 xx xx "ces" $00 xx xx
```

There may be more than one "SYLT" frame in each tag, but only one with the same language and content descriptor.

Comments

This frame is intended for any kind of full text information that does not fit in any other frame. It consists of a frame header followed by encoding, language and content descriptors and is ended with the actual comment as a text string. Newline characters are allowed in the comment text string. There may be more than one comment frame in each tag, but only one with the same language and content descriptor.

```
<Header for 'Comment', ID: "COMM">
Text encoding          $xx
Language               $xx xx xx
Short content descrip. <text string according to encoding> $00 (00)
The actual text        <full text string according to encoding>
```

Relative volume adjustment (2)

This is a more subjective frame than the previous ones. It allows the user to say how much he wants to increase/decrease the volume on each channel when the file is played. The purpose is to be able to align all files to a reference volume, so that you don't have to change the volume constantly. This frame may also be used to balance adjust the audio. The volume adjustment is encoded as a fixed point decibel value, 16 bit signed integer representing (adjustment*512), giving +/- 64 dB with a precision of 0.001953125 dB. E.g. +2 dB is stored as $04 00 and -2 dB is $FC 00. There may be more than one "RVA2" frame in each tag, but only one with the same identification string.

```
<Header for 'Relative volume adjustment (2)', ID: "RVA2">
Identification          <text string> $00
```

The 'identification' string is used to identify the situation and/or device where this adjustment should apply. The following is then repeated for every channel

```
Type of channel         $xx
Volume adjustment       $xx xx
Bits representing peak  $xx
Peak volume             $xx (xx ...)
```

Type of channel:

```
$00  Other
$01  Master volume
$02  Front right
$03  Front left
$04  Back right
$05  Back left
$06  Front centre
```

```
$07  Back centre
$08  Subwoofer
```

Bits representing peak can be any number between 0 and 255. 0 means that there is no peak volume field. The peak volume field is always padded to whole bytes, setting the most significant bits to zero.

Equalisation (2)

This is another subjective, alignment frame. It allows the user to predefine an equalisation curve within the audio file. There may be more than one "EQU2" frame in each tag, but only one with the same identification string.

```
<Header of 'Equalisation (2)', ID: "EQU2">
Interpolation method  $xx
Identification        <text string> $00
```

The 'interpolation method' describes which method is preferred when an interpolation between the adjustment point that follows. The following methods are currently defined:

```
$00  Band
     No interpolation is made. A jump from one adjustment level to
     another occurs in the middle between two adjustment points.
$01  Linear
     Interpolation between adjustment points is linear.
```

The 'identification' string is used to identify the situation and/or device where this adjustment should apply. The following is then repeated for every adjustment point

```
Frequency          $xx xx
Volume adjustment  $xx xx
```

The frequency is stored in units of 1/2 Hz, giving it a range from 0 to 32767 Hz.

The volume adjustment is encoded as a fixed point decibel value, 16 bit signed integer representing (adjustment*512), giving +/- 64 dB with a precision of 0.001953125 dB. E.g. +2 dB is stored as $04 00 and -2 dB is $FC 00.

Adjustment points should be ordered by frequency and one frequency should only be described once in the frame.

Reverb

Yet another subjective frame, with which you can adjust echoes of different kinds. Reverb left/right is the delay between every bounce in ms. Reverb bounces left/right is the number of bounces that should be made. $FF equals an infinite number of bounces. Feedback is the amount of volume that should be returned to the next echo bounce. $00 is 0%, $FF is 100%. If this value were $7F, there would be 50% volume reduction on the first bounce, 50% of that on the second and so on. Left to left means the sound from the left bounce to be played in the left speaker, while left to right means sound from the left bounce to be played in the right speaker.

'Premix left to right' is the amount of left sound to be mixed in the right before any reverb is applied, where $00 id 0% and $FF is 100%. 'Premix right to left' does the same thing, but right to left. Setting both premix to $FF would result in a mono output (if the reverb is applied symmetric). There may only be one "RVRB" frame in each tag.

```
<Header for 'Reverb', ID: "RVRB">
Reverb left (ms)                 $xx xx
Reverb right (ms)                $xx xx
Reverb bounces, left             $xx
Reverb bounces, right            $xx
Reverb feedback, left to left    $xx
Reverb feedback, left to right   $xx
```

```
Reverb feedback, right to right  $xx
Reverb feedback, right to left   $xx
Premix left to right             $xx
Premix right to left             $xx
```

Attached picture

This frame contains a picture directly related to the audio file. Image format is the MIME type and subtype [*MIME*] for the image. In the event that the MIME media type name is omitted, "image/" will be implied. The "image/png" [*PNG*] or "image/jpeg" [*JFIF*] picture format should be used when interoperability is wanted. Description is a short description of the picture, represented as a terminated text string. There may be several pictures attached to one file, each in their individual "APIC" frame, but only one with the same content descriptor. There may only be one picture with the picture type declared as picture type $01 and $02 respectively. There is the possibility to put only a link to the image file by using the 'MIME type' "–>" and having a complete URL [URL] instead of picture data. The use of linked files should however be used sparingly since there is the risk of separation of files.

```
<Header for 'Attached picture', ID: "APIC">
Text encoding      $xx
MIME type          <text string> $00
Picture type       $xx
Description        <text string according to encoding> $00 (00)
Picture data       <binary data>
```

Picture type:

```
$00  Other
$01  32x32 pixels 'file icon' (PNG only)
$02  Other file icon
$03  Cover (front)
$04  Cover (back)
$05  Leaflet page
$06  Media (e.g. label side of CD)
$07  Lead artist/lead performer/soloist
$08  Artist/performer
$09  Conductor
$0A  Band/Orchestra
$0B  Composer
$0C  Lyricist/text writer
$0D  Recording Location
$0E  During recording
$0F  During performance
$10  Movie/video screen capture
$11  A bright coloured fish
$12  Illustration
$13  Band/artist logotype
$14  Publisher/Studio logotype
```

General encapsulated object

In this frame any type of file can be encapsulated. After the header, 'Frame size' and 'Encoding' follows 'MIME type' [MIME] represented as as a terminated string encoded with ISO 8859-1 [ISO-8859-1]. The filename is case sensitive and is encoded as 'Encoding'. Then follows a content description as terminated string, encoded as 'Encoding'. The last thing in the frame is the actual object. The first two strings may be omitted, leaving only their terminations. MIME type is always an ISO-8859-1 text string. There may be more than one "GEOB" frame in each tag, but only one with the same content descriptor.

```
<Header for 'General encapsulated object', ID: "GEOB">
Text encoding          $xx
MIME type              <text string> $00
Filename               <text string according to encoding> $00 (00)
Content description    <text string according to encoding> $00 (00)
Encapsulated object    <binary data>
```

Play counter

This is simply a counter of the number of times a file has been played. The value is increased by one every time the file begins to play. There may only be one "PCNT" frame in each tag. When the counter reaches all one's, one byte is inserted in front of the counter thus making the counter eight bits bigger. The counter must be at least 32-bits long to begin with.

```
<Header for 'Play counter', ID: "PCNT">
Counter        $xx xx xx xx (xx ...)
```

Popularimeter

The purpose of this frame is to specify how good an audio file is. Many interesting applications could be found to this frame such as a playlist that features better audio files more often than others or it could be used to profile a person's taste and find other 'good' files by comparing people's profiles. The frame contains the email address to the user, one rating byte and a four byte play counter, intended to be increased with one for every time the file is played. The email is a terminated string. The rating is 1-255 where 1 is worst and 255 is best. 0 is unknown. If no personal counter is wanted it may be omitted. When the counter reaches all one's, one byte is inserted in front of the counter thus making the counter eight bits bigger in the same away as the play counter ("PCNT"). There may be more than one "POPM" frame in each tag, but only one with the same email address.

```
<Header for 'Popularimeter', ID: "POPM">
Email to user   <text string> $00
Rating          $xx
Counter         $xx xx xx xx (xx ...)
```

Recommended buffer size

Sometimes the server from which an audio file is streamed is aware of transmission or coding problems resulting in interruptions in the audio stream. In these cases, the size of the buffer can be recommended by the server using this frame. If the 'embedded info flag' is true (1) then this indicates that an ID3 tag with the maximum size described in 'Buffer size' may occur in the audio stream. In such case the tag should reside between two MPEG [MPEG] frames, if the audio is MPEG encoded. If the position of the next tag is known, 'offset to next tag' may be used. The offset is calculated from the end of tag in which this frame resides to the first byte of the header in the next. This field may be omitted. Embedded tags are generally not recommended since this could render unpredictable behaviour from present software/hardware.

For applications like streaming audio it might be an idea to embed tags into the audio stream though. If the clients connects to individual connections like HTTP and there is a possibility to begin every transmission with a tag, then this tag should include a 'recommended buffer size' frame. If the client is connected to a arbitrary point in the stream, such as radio or multicast, then the 'recommended buffer size' frame SHOULD be included in every tag.

The 'Buffer size' should be kept to a minimum. There may only be one "RBUF" frame in each tag.

```
<Header for 'Recommended buffer size', ID: "RBUF">
Buffer size               $xx xx xx
```

```
Embedded info flag        %0000000x
Offset to next tag        $xx xx xx xx
```

Audio encryption

This frame indicates if the actual audio stream is encrypted, and by whom. Since standardisation of such encryption scheme is beyond this document, all "AENC" frames begin with a terminated string with a URL containing an email address, or a link to a location where an email address can be found, that belongs to the organisation responsible for this specific encrypted audio file. Questions regarding the encrypted audio should be sent to the email address specified. If a $00 is found directly after the 'Frame size' and the audio file indeed is encrypted, the whole file may be considered useless.

After the 'Owner identifier', a pointer to an unencrypted part of the audio can be specified. The 'Preview start' and 'Preview length' is described in frames. If no part is unencrypted, these fields should be left zeroed. After the 'preview length' field follows optionally a data block required for decryption of the audio. There may be more than one "AENC" frames in a tag, but only one with the same 'Owner identifier'.

```
<Header for 'Audio encryption', ID: "AENC">
Owner identifier   <text string> $00
Preview start      $xx xx
Preview length     $xx xx
Encryption info    <binary data>
```

Linked information

To keep information duplication as low as possible this frame may be used to link information from another ID3v2 tag that might reside in another audio file or alone in a binary file. It is RECOMMENDED that this method is only used when the files are stored on a CD-ROM or other circumstances when the risk of file separation is low. The frame contains a frame identifier, which is the frame that should be linked into this tag, a URL [URL] field, where a reference to the file where the frame is given, and additional ID data, if needed. Data should be retrieved from the first tag found in the file to which this link points. There may be more than one "LINK" frame in a tag, but only one with the same contents. A linked frame is to be considered as part of the tag and has the same restrictions as if it was a physical part of the tag (i.e. only one "RVRB" frame allowed, whether it's linked or not).

```
<Header for 'Linked information', ID: "LINK">
Frame identifier        $xx xx xx xx
URL                     <text string> $00
ID and additional data  <text string(s)>
```

Frames that may be linked and need no additional data are "ASPI", "ETCO", "EQU2", "MCID", "MLLT", "OWNE", "RVA2", "RVRB", "SYTC", the text information frames and the URL link frames.

The "AENC", "APIC", "GEOB" and "TXXX" frames may be linked with the content descriptor as additional ID data.

The "USER" frame may be linked with the language field as additional ID data.

The "PRIV" frame may be linked with the owner identifier as additional ID data.

The "COMM", "SYLT" and "USLT" frames may be linked with three bytes of language descriptor directly followed by a content descriptor as additional ID data.

Position synchronisation frame

This frame delivers information to the listener of how far into the audio stream he picked up; in effect, it states the time offset from the first frame in the stream. The frame layout is:

```
<Head for 'Position synchronisation', ID: "POSS">
Time stamp format         $xx
Position                  $xx (xx ...)
```

Where time stamp format is:

```
$01  Absolute time, 32 bit sized, using MPEG frames as unit
$02  Absolute time, 32 bit sized, using milliseconds as unit
```

and position is where in the audio the listener starts to receive, i.e. the beginning of the next frame. If this frame is used in the beginning of a file the value is always 0. There may only be one "POSS" frame in each tag.

Terms of use frame

This frame contains a brief description of the terms of use and ownership of the file. More detailed information concerning the legal terms might be available through the "WCOP" frame. Newlines are allowed in the text. There may be more than one 'Terms of use' frame in a tag, but only one with the same 'Language'.

```
<Header for 'Terms of use frame', ID: "USER">
Text encoding        $xx
Language             $xx xx xx
The actual text      <text string according to encoding>
```

Ownership frame

The ownership frame might be used as a reminder of a made transaction or, if signed, as proof. Note that the "USER" and "TOWN" frames are good to use in conjunction with this one. The frame begins, after the frame ID, size and encoding fields, with a 'price paid' field. The first three characters of this field contains the currency used for the transaction, encoded according to ISO 4217 [ISO-4217] alphabetic currency code. Concatenated to this is the actual price paid, as a numerical string using "." as the decimal separator. Next is an 8 character date string (YYYYMMDD) followed by a string with the name of the seller as the last field in the frame. There may only be one "OWNE" frame in a tag.

```
<Header for 'Ownership frame', ID: "OWNE">
Text encoding     $xx
Price paid        <text string> $00
Date of purch.    <text string>
Seller            <text string according to encoding>
```

Commercial frame

This frame enables several competing offers in the same tag by bundling all needed information. That makes this frame rather complex but it's an easier solution than if one tries to achieve the same result with several frames. The frame begins, after the frame ID, size and encoding fields, with a price string field. A price is constructed by one three character currency code, encoded according to ISO 4217 [ISO-4217] alphabetic currency code, followed by a numerical value where "." is used as decimal separator. In the price string several prices may be concatenated, separated by a "/" character, but there may only be one currency of each type.

The price string is followed by an 8 character date string in the format YYYYMMDD, describing for how long the price is valid. After that is a contact URL, with which the user can contact the seller, followed by a one byte 'received as' field. It describes how the audio is delivered when bought according to the following list:

```
$00  Other
$01  Standard CD album with other songs
$02  Compressed audio on CD
$03  File over the Internet
$04  Stream over the Internet
$05  As note sheets
$06  As note sheets in a book with other sheets
$07  Music on other media
$08  Non-musical merchandise
```

Next follows a terminated string with the name of the seller followed by a terminated string with a short description of the product. The last thing is the ability to include a company logotype. The first of them is the 'Picture MIME type' field containing information about which picture format is used. In the event that the MIME media type name is omitted, "image/" will be implied. Currently only "image/png" and "image/jpeg" are allowed. This format string is followed by the binary picture data. This two last fields may be omitted if no picture is attached. There may be more than one 'commercial frame' in a tag, but no two may be identical.

```
<Header for 'Commercial frame', ID: "COMR">
Text encoding      $xx
Price string       <text string> $00
Valid until        <text string>
Contact URL        <text string> $00
Received as        $xx
Name of seller     <text string according to encoding> $00 (00)
Description        <text string according to encoding> $00 (00)
Picture MIME type  <string> $00
Seller logo        <binary data>
```

Encryption method registration

To identify with which method a frame has been encrypted the encryption method must be registered in the tag with this frame. The 'Owner identifier' is a null-terminated string with a URL [URL] containing an email address, or a link to a location where an email address can be found, that belongs to the organisation responsible for this specific encryption method. Questions regarding the encryption method should be sent to the indicated email address. The 'Method symbol' contains a value that is associated with this method throughout the whole tag, in the range $80-F0. All other values are reserved. The 'Method symbol' may optionally be followed by encryption specific data. There may be several "ENCR" frames in a tag but only one containing the same symbol and only one containing the same owner identifier. The method must be used somewhere in the tag. See the description of the frame encryption flag in the ID3v2 structure document [ID3v2-strct] for more information.

```
<Header for 'Encryption method registration', ID: "ENCR">
Owner identifier    <text string> $00
Method symbol       $xx
Encryption data     <binary data>
```

Group identification registration

This frame enables grouping of otherwise unrelated frames. This can be used when some frames are to be signed. To identify which frames belongs to a set of frames a group identifier must be registered in the tag with this frame. The 'Owner identifier' is a null-terminated string with a URL [URL] containing an email address, or a link to a location where an email address can be found, that belongs to the organisation responsible for this grouping. Questions regarding the grouping should be sent to the indicated email address. The 'Group symbol' contains a value that associates the frame with this group throughout the

whole tag, in the range $80-F0. All other values are reserved. The 'Group symbol' may optionally be followed by some group specific data, e.g. a digital signature. There may be several "GRID" frames in a tag but only one containing the same symbol and only one containing the same owner identifier. The group symbol must be used somewhere in the tag. See the description of the frame grouping flag in the ID3v2 structure document [ID3v2-strct] for more information.

```
<Header for 'Group ID registration', ID: "GRID">
Owner identifier      <text string> $00
Group symbol          $xx
Group dependent data  <binary data>
```

Private frame

This frame is used to contain information from a software producer that its program uses and does not fit into the other frames. The frame consists of an 'Owner identifier' string and the binary data. The 'Owner identifier' is a null-terminated string with a URL [URL] containing an email address, or a link to a location where an email address can be found, that belongs to the organisation responsible for the frame. Questions regarding the frame should be sent to the indicated email address. The tag may contain more than one "PRIV" frame but only with different contents.

```
<Header for 'Private frame', ID: "PRIV">
Owner identifier      <text string> $00
The private data      <binary data>
```

Signature frame

This frame enables a group of frames, grouped with the 'Group identification registration', to be signed. Although signatures can reside inside the registration frame, it might be desired to store the signature elsewhere, e.g. in watermarks. There may be more than one 'signature frame' in a tag, but no two may be identical.

```
<Header for 'Signature frame', ID: "SIGN">
Group symbol      $xx
Signature         <binary data>
```

Seek frame

This frame indicates where other tags in a file/stream can be found. The 'minimum offset to next tag' is calculated from the end of this tag to the beginning of the next. There may only be one 'seek frame' in a tag.

```
<Header for 'Seek frame', ID: "SEEK">
Minimum offset to next tag       $xx xx xx xx
```

Audio seek point index

Audio files with variable bit rates are intrinsically difficult to deal with in the case of seeking within the file. The ASPI frame makes seeking easier by providing a list a seek points within the audio file. The seek points are a fractional offset within the audio data, providing a starting point from which to find an appropriate point to start decoding. The presence of an ASPI frame requires the existence of a TLEN frame, indicating the duration of the file in milliseconds. There may only be one 'audio seek point index' frame in a tag.

```
<Header for 'Seek Point Index', ID: "ASPI">
Indexed data start (S)          $xx xx xx xx
Indexed data length (L)         $xx xx xx xx
Number of index points (N)      $xx xx
Bits per index point (b)        $xx
```

Then for every index point the following data is included;

```
Fraction at index (Fi)           $xx (xx)
```

'Indexed data start' is a byte offset from the beginning of the file. 'Indexed data length' is the byte length of the audio data being indexed. 'Number of index points' is the number of index points, as the name implies. The recommended number is 100. 'Bits per index point' is 8 or 16, depending on the chosen precision. 8 bits works well for short files (less than 5 minutes of audio), while 16 bits is advantageous for long files. 'Fraction at index' is the numerator of the fraction representing a relative position in the data. The denominator is 2 to the power of b.

Here are the algorithms to be used in the calculation. The known data must be the offset of the start of the indexed data (S), the offset of the end of the indexed data (E), the number of index points (N), the offset at index i (Oi). We calculate the fraction at index i (Fi).

Oi is the offset of the frame whose start is soonest after the point for which the time offset is (i/N * duration).

The frame data should be calculated as follows:

```
Fi = Oi/L * 2^b     (rounded down to the nearest integer)
```

Offset calculation should be calculated as follows from data in the frame:

```
Oi = (Fi/2^b)*L     (rounded up to the nearest integer)
```

1.4.6 Copyright

1.4.7 References

CDDB Compact Disc Data Base

ID3v2.3.0 Martin Nilsson, ID3v2 informal standard

ID3v2-strct Martin Nilsson, ID3 tag version 2.4.0 - Main Structure

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-4217 ISO 4217:1995. Codes for the representation of currencies and funds. Technical committee / subcommittee: TC 68

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

ISRC ISO 3901:1986 International Standard Recording Code (ISRC). Technical committee / subcommittee: TC 46 / SC 9

JFIF JPEG File Interchange Format, version 1.02

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

MIME Freed, N. and N. Borenstein, Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies RFC 2045, November 1996.

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 T. Berners-Lee, L. Masinter & M. McCahill, Uniform Resource Locators (URL)., RFC 1738, December 1994.

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

1.4.8 Appendix

Appendix A - Genre List from ID3v1

The following genres is defined in ID3v1

0. Blues
1. Classic Rock
2. Country
3. Dance
4. Disco
5. Funk
6. Grunge
7. Hip-Hop
8. Jazz
9. Metal
10. New Age
11. Oldies

12. Other
13. Pop
14. R&B
15. Rap
16. Reggae
17. Rock
18. Techno
19. Industrial
20. Alternative
21. Ska
22. Death Metal
23. Pranks
24. Soundtrack
25. Euro-Techno
26. Ambient
27. Trip-Hop
28. Vocal
29. Jazz+Funk
30. Fusion
31. Trance
32. Classical
33. Instrumental
34. Acid
35. House
36. Game
37. Sound Clip
38. Gospel
39. Noise
40. AlternRock
41. Bass
42. Soul
43. Punk
44. Space
45. Meditative
46. Instrumental Pop
47. Instrumental Rock

48. Ethnic
49. Gothic
50. Darkwave
51. Techno-Industrial
52. Electronic
53. Pop-Folk
54. Eurodance
55. Dream
56. Southern Rock
57. Comedy
58. Cult
59. Gangsta
60. Top 40
61. Christian Rap
62. Pop/Funk
63. Jungle
64. Native American
65. Cabaret
66. New Wave
67. Psychadelic
68. Rave
69. Showtunes
70. Trailer
71. Lo-Fi
72. Tribal
73. Acid Punk
74. Acid Jazz
75. Polka
76. Retro
77. Musical
78. Rock & Roll
79. Hard Rock

1.4.9 Author's Address

Written by

Martin Nilsson
Rydsvägen 246 C. 30
SE-584 34 Linköping
Sweden

Email: nilsson at id3.org

1.5 ID3v2 Chapters 1.0

1.5.1 Status of this document

This document is an addendum to the ID3v2.3 and ID3v2.4 standards. Distribution of this document is unlimited.

1.5.2 Abstract

This document describes a method for signalling chapters and a table of contents within an audio file using two new ID3v2 frames. The frames allow listeners to navigate to specific locations in an audio file and can provide descriptive information, URLs and images related to each chapter.

1.5.3 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 the ID3v2 main structure document [v2.4]. 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].

1.5.4 Declared ID3v2 frames

Chapter frame

The purpose of this frame is to describe a single chapter within an audio file. There may be more than one frame of this type in a tag but each must have an Element ID that is unique with respect to any other "CHAP" frame or "CTOC" frame in the tag.

```
<ID3v2.3 or ID3v2.4 frame header, ID: "CHAP">           (10 bytes)
Element ID      <text string> $00
Start time      $xx xx xx xx
End time        $xx xx xx xx
Start offset    $xx xx xx xx
End offset      $xx xx xx xx
<Optional embedded sub-frames>
```

The Element ID uniquely identifies the frame. It is not intended to be human readable and should not be presented to the end user.

The Start and End times are a count in milliseconds from the beginning of the file to the start and end of the chapter respectively.

The Start offset is a zero-based count of bytes from the beginning of the file to the first byte of the first audio frame in the chapter. If these bytes are all set to 0xFF then the value should be ignored and the start time value should be utilized.

The End offset is a zero-based count of bytes from the beginning of the file to the first byte of the audio frame following the end of the chapter. If these bytes are all set to 0xFF then the value should be ignored and the end time value should be utilized.

There then follows a sequence of optional frames that are embedded within the "CHAP" frame and which describe the content of the chapter (e.g. a "TIT2" frame representing the chapter name) or provide related material such as URLs and images. These sub-frames are contained within the bounds of the "CHAP" frame as signalled by the size field in the "CHAP" frame header. If a parser does not recognise "CHAP" frames it can skip them using the size field in the frame header. When it does this it will skip any embedded sub-frames carried within the frame.

Figure 1.1 shows an example of a "CHAP" frame containing two embedded sub-frames. The first is a "TIT2" sub-frame providing the chapter name; "Chapter 1 - Loomings". The second is a "TIT3" sub-frame providing a description of the chapter; "Anticipation of the hunt".

Table of contents frame

The purpose of "CTOC" frames is to allow a table of contents to be defined. In the simplest case, a single "CTOC" frame can be used to provide a flat (single-level) table of contents. However, multiple "CTOC" frames can also be used to define a hierarchical (multi-level) table of contents.

There may be more than one frame of this type in a tag but each must have an Element ID that is unique with respect to any other "CTOC" or "CHAP" frame in the tag.

Each "CTOC" frame represents one level or element of a table of contents by providing a list of Child Element IDs. These match the Element IDs of other "CHAP" and "CTOC" frames in the tag.

```
<ID3v2.3 or ID3v2.4 frame header, ID: "CTOC">   (10 bytes)
Element ID      <text string> $00
Flags           %000000ab
Entry count     $xx  (8-bit unsigned int)
<Child Element ID list>
<Optional embedded sub-frames>
```

The Element ID uniquely identifies the frame. It is not intended to be human readable and should not be presented to the end-user.

Flag a - Top-level bit This is set to 1 to identify the top-level "CTOC" frame. This frame is the root of the Table of Contents tree and is not a child of any other "CTOC" frame. Only one "CTOC" frame in an ID3v2 tag can have this bit set to 1. In all other "CTOC" frames this bit shall be set to 0.

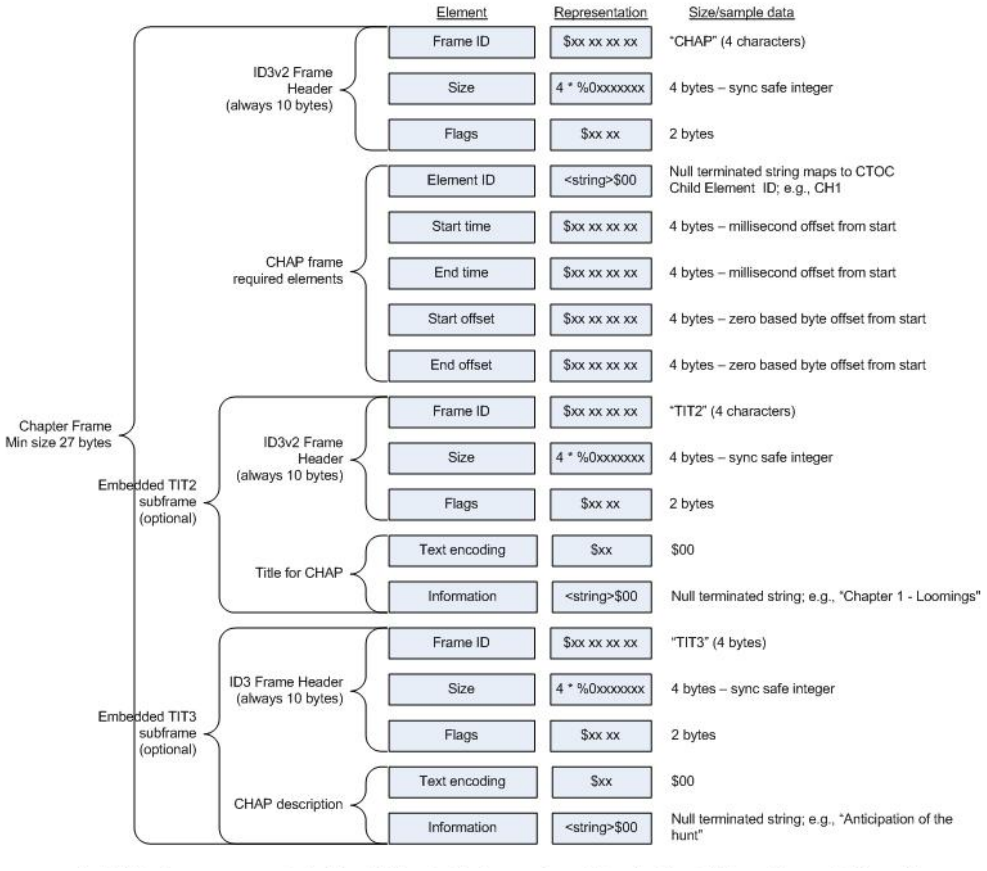

Fig. 1.1: Example CHAP frame

Flag b - Ordered bit This should be set to 1 if the entries in the Child Element ID list are ordered or set to 0 if they not are ordered. This provides a hint as to whether the elements should be played as a continuous ordered sequence or played individually. The Entry count is the number of entries in the Child Element ID list that follows and must be greater than zero. Each entry in the list consists of:

```
Child Element ID           <text string> $00
```

The last entry in the child Element ID list is followed by a sequence of optional frames that are embedded within the "CTOC" frame and which describe this element of the table of contents (e.g. a "TIT2" frame representing the name of the element) or provide related material such as URLs and images. These sub-frames are contained within the bounds of the "CTOC" frame as signalled by the size field in the "CTOC" frame header.

If a parser does not recognise "CTOC" frames it can skip them using the size field in the frame header. When it does this it will skip any embedded sub-frames carried within the frame.

Figure 1.2 shows an example of a "CTOC" frame which references a sequence of chapters. It contains a single "TIT2" sub-frame which provides a name for this element of the table of contents; "Part 1".

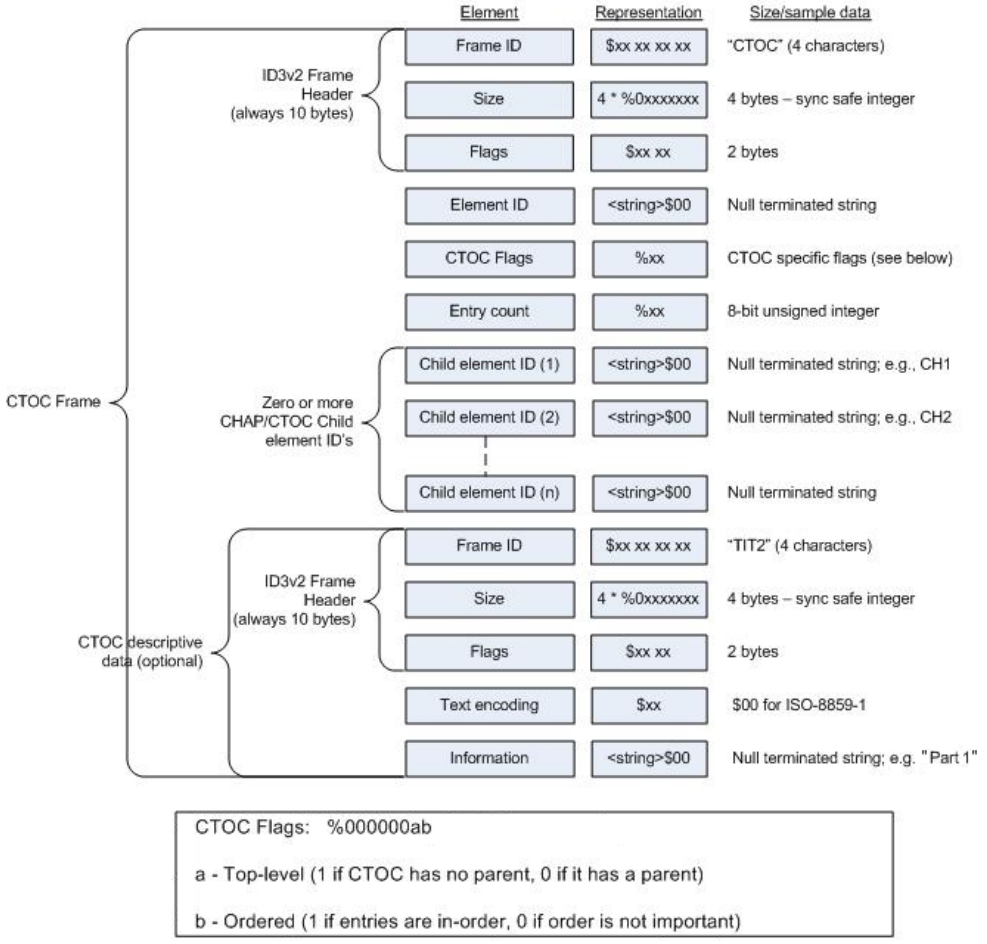

Fig. 1.2: Example CTOC frame

1.5.5 Notes

- It is possible for "CHAP" frames to describe chapters that overlap or have gaps between them.

- It is permitted to include "CHAP" frames that are not referenced by any "CTOC" frames. For example, these might be used to provide images that can be presented in synchronisation with the audio, rather than to support a table of contents.
- It is recommended that "CHAP" and "CTOC" frames should include a TIT2 sub-frame to provide a human readable identifier which can be presented to the end-user to aid navigation and selection.

1.5.6 Copyright

1.5.7 References

[v2.3] Martin Nilsson, ID3 tag version 2.3.0.

[v2.4] Martin Nilsson, ID3 tag version 2.4.0 - Main Structure.

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

1.5.8 Author's Address

Chris Newell
BBC Research & Development
Kingswood Warren
Tadworth
Surrey
KT20 6NP
UK

Email: chris.newell at rd.bbc.co.uk

1.6 ID3v2 Accessibility 1.0

1.6.1 Status of this document

This document is a proposed addendum to the ID3v2.3 and ID3v2.4 standards. Distribution of this document is unlimited.

1.6.2 Abstract

This document describes extensions which make ID3v2 metadata accessible to the visually impaired. The approach may also be useful for audio players which have limited display capabilities. A new frame type is proposed that carries an audio clip which can provide a verbal expression of the textual information carried by another ID3v2 frame.

1.6.3 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 the ID3v2 main structure document [2]. 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.

1.6.4 Introduction

The ID3v2 standards provide a way to deliver metadata that is predominantly human-readable, textual data. However, in this form the information is not easily accessible to the visually impaired.

The purpose of this Addendum is to allow content providers or third-party tools to provide an audio description (i.e. a spoken narrative) that is equivalent to the textual information carried by an ID3v2 frame. A new "audio-text" frame is defined which carries an audio clip and a matching equivalent text string. These text strings can be compared against the strings carried by other ID3v2 frames to identify when a matching audio description is available.

The audio clips can be played whenever the equivalent textual information is displayed or highlighted, providing a greatly improved user interface for the visually impaired. However, the feature may also be popular with other users and useful for media players with limited display capabilities.

1.6.5 Proposed audio-text frame

The purpose of this frame is to carry a short audio clip which represents the information carried by another ID3v2 frame that is present in the same tag.

To avoid these audio clips being confused with the main audio content of the file the ID3v2 unsynchronisation scheme must be used if the audio clip uses an MPEG audio format. If the unsynchronisation scheme is not appropriate for the audio format then the scrambling scheme defined in section 5 must be applied to the audio clip data.

```
<ID3v2.3 or ID3v2.4 frame header, ID: "ATXT">
Text encoding   $xx
MIME type       <text string> $00
Flags           %0000000a
Equivalent text <text string according to encoding> $00 (00)
Audio data      <binary data>
```

The Frame ID for the audio-text frame shall be set to "ATXT" using ISO-8859-1 character encoding.

The MIME type shall be represented as a terminated string encoded using ISO-8859-1 character encoding. Where the MIME type corresponds to MPEG 1/2 layer I, II and III, MPEG 2.5 or AAC audio the ID3v2 unsynchronisation

scheme should be applied, either to the audio-text frame or to the tag which contains it. For other MIME types the scrambling scheme defined in the Appendix should be applied to the audio data.

Flag a - Scrambling flag This flag shall be set if the scrambling method defined in Section 5 has been applied to the audio data, or not set if no scrambling has been applied.

The Equivalent text field carries a null terminated string encoded according to the Text encoding byte as defined by the ID3v2 specifications [1], [2]. This text must be semantically equivalent to the spoken narrative in the audio clip and should match the text and encoding used by another ID3v2 frame in the tag.

The Audio data carries an audio clip which provides the audio description. The encoding of the audio data shall match the MIME type field and the data shall be scrambled if the scrambling flag is set.

More than one audio-text frame may be present in a tag but each must carry a unique string in the Equivalent text field.

1.6.6 Scrambling scheme for non-MPEG audio formats

This scrambling scheme is provided for non-MPEG audio formats where the unsynchronisation scheme defined by the ID3v2 specifications is unsuitable. Each bit of the audio data is scrambled by taking the exclusive-OR (XOR) between it and the equivalent bit of a pseudo-random byte sequence. The first byte of this pseudo-random byte sequence is always %11111110 and is used to scramble the first byte of the audio data. The next byte of the sequence is derived from the current byte of the sequence using the algorithm in Table 1 and is used to scramble the next byte of audio data. This process is repeated until all bytes in the audio clip have been scrambled.

Table 1: Scrambling sequence algorithm

byte N+1	byte N
~~bit 7 =~~	~~bit 6 XOR bit 5~~
bit 6 =	bit 5 XOR bit 4
~~bit 5 =~~	~~bit 4 XOR bit 3~~
bit 4 =	bit 3 XOR bit 2
~~bit 3 =~~	~~bit 2 XOR bit 1~~
bit 2 =	bit 1 XOR bit 0
~~bit 1 =~~	~~bit 7 XOR bit 5~~
bit 0 =	bit 6 XOR bit 4

This algorithm results in a 127-bit pseudo-random sequence which repeats on byte boundaries every 127 bytes. To recover the audio data from the scrambled data the scrambling procedure is repeated.

1.6.7 Notes

- Failure to use the ID3v2 unsynchronisation scheme or the alternative scrambling scheme, as appropriate to the audio format, is very likely to confuse media players which are likely to start playback when an audio-text frame in encountered rather than at the end of the ID3v2 tag.
- Players which only support MPEG audio formats are not required to support the scrambling scheme provided for non-MPEG formats.
- It is not required to provide an audio-text frame to represent every text string present in a tag. The emphasis should be on text strings in frames that are commonly used to identify and describe the content (e.g "*TIT2*", "*TALB*" & "*TPE1*").
- A parser that does not recognise "ATXT" frames can skip them using the size field in the frame header.
- Editing text fields in ID3 tags may result in the retention of irrelevant ATXT frames and gaps in the provision of audio text unless action is taken to amend the corresponding ATXT frames.

1.6.8 Copyright

1.6.9 References

Martin Nilsson, ID3 tag version 2.3.0.

Martin Nilsson, ID3 tag version 2.4.0 - Main Structure.

13. Nilsson, "ID3 tag version 2.4.0 - Native frames.

S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, March 1997.

1.6.10 Author's address

Chris Newell
BBC Research & Development
Kingswood Warren
Tadworth
Surrey
KT20 6NP
UK

Email: chris.newell at bbc.co.uk

1.7 ID3v1 Winamp Genre Mapping

ID	Name
0	Blues
1	Classic Rock
2	Country
3	Dance
4	Disco
5	Funk
6	Grunge
7	Hip-Hop
8	Jazz
9	Metal
10	New Age
11	Oldies
12	Other
13	Pop
14	R&B
15	Rap
16	Reggae
17	Rock
18	Techno
19	Industrial
20	Alternative
21	Ska
22	Death Metal
23	Pranks
24	Soundtrack
25	Euro-Techno
26	Ambient
27	Trip-Hop
28	Vocal
29	Jazz+Funk
30	Fusion
31	Trance
32	Classical
33	Instrumental
34	Acid
35	House
36	Game
37	Sound Clip
38	Gospel
39	Noise
40	Alt. Rock
41	Bass
42	Soul
43	Punk

Continued on next page

Table 1.1 – continued from previous page

ID	Name
44	Space
45	Meditative
46	Instrumental Pop
47	Instrumental Rock
48	Ethnic
49	Gothic
50	Darkwave
51	Techno-Industrial
52	Electronic
53	Pop-Folk
54	Eurodance
55	Dream
56	Southern Rock
57	Comedy
58	Cult
59	Gangsta Rap
60	Top 40
61	Christian Rap
62	Pop/Funk
63	Jungle
64	Native American
65	Cabaret
66	New Wave
67	Psychedelic
68	Rave
69	Showtunes
70	Trailer
71	Lo-Fi
72	Tribal
73	Acid Punk
74	Acid Jazz
75	Polka
76	Retro
77	Musical
78	Rock & Roll
79	Hard Rock
80	Folk
81	Folk-Rock
82	National Folk
83	Swing
84	Fast-Fusion
85	Bebop
86	Latin
87	Revival
88	Celtic
89	Bluegrass
90	Avantgarde
91	Gothic Rock
92	Progressive Rock

Continued on next page

Table 1.1 – continued from previous page

ID	Name
93	Psychedelic Rock
94	Symphonic Rock
95	Slow Rock
96	Big Band
97	Chorus
98	Easy Listening
99	Acoustic
100	Humour
101	Speech
102	Chanson
103	Opera
104	Chamber Music
105	Sonata
106	Symphony
107	Booty Bass
108	Primus
109	Porn Groove
110	Satire
111	Slow Jam
112	Club
113	Tango
114	Samba
115	Folklore
116	Ballad
117	Power Ballad
118	Rhythmic Soul
119	Freestyle
120	Duet
121	Punk Rock
122	Drum Solo
123	A Cappella
124	Euro-House
125	Dance Hall
126	Goa
127	Drum & Bass
128	Club-House
129	Hardcore
130	Terror
131	Indie
132	BritPop
133	Afro-Punk
134	Polsk Punk
135	Beat
136	Christian Gangsta Rap
137	Heavy Metal
138	Black Metal
139	Crossover
140	Contemporary Christian
141	Christian Rock

Continued on next page

Table 1.1 – continued from previous page

ID	Name
142	Merengue
143	Salsa
144	Thrash Metal
145	Anime
146	JPop
147	Synthpop
148	Abstract
149	Art Rock
150	Baroque
151	Bhangra
152	Big Beat
153	Breakbeat
154	Chillout
155	Downtempo
156	Dub
157	EBM
158	Eclectic
159	Electro
160	Electroclash
161	Emo
162	Experimental
163	Garage
164	Global
165	IDM
166	Illbient
167	Industro-Goth
168	Jam Band
169	Krautrock
170	Leftfield
171	Lounge
172	Math Rock
173	New Romantic
174	Nu-Breakz
175	Post-Punk
176	Post-Rock
177	Psytrance
178	Shoegaze
179	Space Rock
180	Trop Rock
181	World Music
182	Neoclassical
183	Audiobook
184	Audio Theatre
185	Neue Deutsche Welle
186	Podcast
187	Indie Rock
188	G-Funk
189	Dubstep
190	Garage Rock

Continued on next page

Table 1.1 – continued from previous page

ID	Name
191	Psybient

1.8 Off-Spec Frames

1.8.1 iTunes

TCMP/TCP - iTunes Compilation Flag
TDES - iTunes Podcast Description
TGID - iTunes Podcast Identifier
TSO2 - iTunes Album Artist Sort
TSOC - iTunes Composer Sort
WFED - iTunes Podcast Feed

- MP3/ID3
  - **MPEG audio header format**, and the Xing VBR header
  - ID3v2.4 structure, ID3v2.4 frame list, ID3v2.3, ID3v2.2, and ID3v1
  - Lyrics3v2

CHAPTER 2

Musepack

2.1 SV8 Specification

Note: All fields, unless explicitly specified otherwise are read and written in Big-Endian order.

2.1.1 Status of this document

This document is a work in progress, but part of the document will not be changed.

Here is how the status is indicated for each part:

Status	Meaning
[final]	This part is final and will not be changed
[beta]	This part will not be changed unless necessary
[alpha]	This part is still discussed and will probably change
nothing specified	This part is here for discussion, and not part of the specification

2.1.2 File magic number

[final] Magic number is on 32bits and is equal to 'MPCK' or 0x4D50434B

2.1.3 File extension

[final] The prefered file extension for musepack files is .mpc

2.1.4 Packet formatting

[final] All packets are formatted using Key / Size / Payload. Keys are 16 bits long. It's equivalent to the packet ID or type. Size is a variable-size field:

```
bits, big-endian
0xxx xxxx                                       - value 0 to  2^7-1
1xxx xxxx  0xxx xxxx                            - value 0 to 2^14-1
1xxx xxxx  1xxx xxxx  0xxx xxxx                 - value 0 to 2^21-1
1xxx xxxx  1xxx xxxx  1xxx xxxx  0xxx xxxx      - value 0 to 2^28-1
...
```

Size defines the packet length in bytes, including the Key and Size fields. So the minimum length of a block is 3 bytes. The payload is the actual packet data. Its size can be null. All unused bits in a packet MUST be null.

Field	Size (bits)	Value
Key	16	"EX"
Size	n*8; 0 < n < 10	0x1A
Payload	Size * 8	"example"

2.1.5 Summary of reserved packet keys

Allowed chars in key are [A-Z] (65 <= value <= 90), so 676 keys are valid out of 65536 possible.

Packet Name	Key	Mandatory	Status
Stream Header	SH	yes	[final]
Replaygain	RG	yes	[final]
Encoder Info	EI	no	[final]
Seek Table Offset	SO	no	[final]
Audio Packet	AP	yes	[final]
Seek Table	ST	no	[final]
Chapter-Tag	CT	no	[beta]
Stream End	SE	yes	[final]

2.1.6 Stream Header Packet

[final] This packet key is "SH". It contains the information needed to decode the stream. This block is mandatory and must be written before the first audio packet.

Field	Size (bits)	Value	Comment
CRC	32		CRC 32 of the block (this field excluded). 0 = invalid
Stream version	8	8	Bitstream version
Sample count	n*8; 0<n<10		Number of samples in the stream. 0 = unknown
Beginning silence	n*8; 0<n<10		Number of samples to skip at the beginning of the stream
Sample frequency	3	0..7	See table below
Max used bands	5	1..32	Maximum number of bands used in the file
Channel count	4	1..16	Number of channels in the stream
MS used	1		True if Mid Side Stereo is enabled
Audio block frames	3	0..7	Number of frames per audio packet (4^value=(1..16384))

Do we need to specify the channel position ? There seems to be no standard for positions / order of the channels.

The SV8 stream allows to start decoding only on the first packet frame, so cutting the stream requires cutting on packet boundaries. Beginning silence and sample count fields provide more precise cutting possibilities.

Frequency table

Value	Frequency (Hz)
0	44100
1	48000
2	37800
3	32000

The CRC used is this one : http://www.w3.org/TR/PNG/#D-CRCAppendixhttp://www.w3.org/TR/PNG/#D-CRCAppendix.

2.1.7 Replaygain Packet

[final] This packet key is "RG". It contains the necessary data needed to apply replaygain on the current stream. This packet is mandatory and must be written before the first audio packet.

Field	Size (bits)	Value	Comment
ReplayGain version	8	1	The replay gain version
Title gain	16		The loudness calculated for the title, and not the gain that the player must apply
Title peak	16		
Album gain	16		The loudness calculated for the album
Album peak	16		

The replay gain values are stored in dB in Q8.8 format. The 0 value means that this field has not been computed (no gain must be applied in this case. exemples :

- Replay gain finds that this title has a loudness of 78.56 dB. It will be encoded as 78.56 * 256 ~ 20111 = 0x4E8F
- For 16-bit output (range [-32767 32768]), the max is 68813 (out of range). It will be encoded as 20 * log10(68813) * 256 ~ 24769 = 0x60C1
- For float output (range [-1 1]), the max is 0.96. It will be encoded as 20 * log10(0.96 * 2^15^) * 256 ~ 23029 = 0x59F5 (for peak values it is suggested to round to nearest higher integer)

2.1.8 Encoder Info Packet

[final] This packet key is "EI".

Field	Size (bits)	Value	Comment
Profile	7	0..15.875	quality in 4.3 format
PNS tool	1	True if enabled	
Major	8	1	Major version
Minor	8	17	Minor version, even numbers for stable version, odd when unstable
Build	8	3	Build

2.1.9 Seek Table Offset Packet

[final] This packet key is "SO". It contains an offset to the seek table packet. This packet must be written before the first audio packet. This packet must be present if the "ST" packet is present and is written after the first audio packet.

Field	Size (bits)	Value	Comment
Offset	n*8; 0 < n < 10		Offset from this packet to the seek table packet

2.1.10 Audio Packet

[final] This packet key is "AP". It contains audio frames. The first frame is a key frame.

Field	Size (bits)	Comment
Audio Frames	?	n (or less if last packet) frames of audio as defined in SH packet

2.1.11 Seek Table Packet

[final] This packet key is "ST".

Field	Size (bits)	Value	Comment
Seek Count	n*8; 0 < n < 10		Number of seek elements in this table
Seek Distance	4	0..15	Distance between referenced blocks = 2^Value
Seek Data	?		

Format of seek data:

- Reference offset for seeking is the musepack magic number
- First 2 values are stored using the same code as the packet size code.
- Next values are coded as:

```
code = value(n) - 2*value(n-1) + value(n-2)
code <<= 1;
if (code < 0)
  code = -code | 1;
```

code is sent as golomb code with M = 2^12^.

2.1.12 Chapter-Tag Packet

[beta] This packet key is "CT". It contains a chapter position and associated tag. There is 1 packet for each chapter. When used in a file, all CT packets must be consecutive. They must be the next (group of) packet after Seek Table packet if present at the end of the file, or the last (group of) packet before the Stream End packet else. Chapters are presented by the application in the same order as they appear in the file. When used while streaming, this packet can be inserted between AP packets, and the tag data is valid for the next samples, until a new CT packet is sent.

Field	Size (bits)	Value	Comment
Sample offset	n*8; 0 < n < 10		Position of the chapter in samples. In a file from the beginning of the file, in a stream from the last sample before this packet
Chapter gain	16		The loudness calculated for the chapter, and not the gain that the player must apply
Chapter peak	16		
APEv2 tag	n*8		APEv2 tag without the preamble { 'A', 'P', 'E', 'T', 'A', 'G', 'E', 'X' } in the header or footer, preferably without footer. This field is optional.

2.1.13 Security Packet

Checksum (MD5, SHA1) or error correcting code (LDPC). To be defined later. May be better to keep security features external only.

2.1.14 Stream End Packet

[final] This packet key is "SE". The packet size must be 3 bytes. This packet is mandatory and must be the last stream packet. Tags, if present, must be written after this packet.

2.1.15 Streaming

[alpha] This file format can be used for streaming. The "SH" block is (can be?) used as a synchronization marker. The decoder will scan for a "SH" block and check its CRC. Once the decoder is synchronized, it will start decoding. It's up to the streaming server to choose when to send "SH" block. To send meta data while streaming, the Chapter-Tag packet can be used.

2.1.16 Tags

[final] No packet must be written after the stream end packet, to allow tagging by other applications. Those tags are global to the file, and define the default values for all the tag fields. The Chapter-Tag packet can redefine the field value for each chapter.

2.1.17 Example file

Packets keys and magic number are highlighted:

```
00000000   4D 50 43 4B  53 48 0F 12  A5 AB 62 08  84 FA C1 40  00 1B 1B 52  MPCKSH....b....@...R
00000014   47 0C 01 00  00 00 00 00  00 00 00 45  49 07 A0 01  17 00 53 4F  G..........EI.....SO
00000028   08 82 C2 83  31 00 41 50  82 B2 01 BA  A7 36 59 FE  BC 7B CD 3E  ....1.AP.....6Y..{.>
0000003C   10 3B EF 9B  3A 8E DA 22  0B 64 9A 67  AE EC 99 CB  2A 66 4C 79  .;..:.."d.g....*fLy

... Lots of Audio Packets

005081C4   BF FF FF 85  03 81 B7 32  A0 2E 3A E0  D4 FC 20 16  A0 40 80 53  .......2..:... ..@.S
005081D8   54 81 01 47  12 E8 58 21  73 01 83 DC  A8 1D 95 4B  D9 F5 37 EF  T..G..X!s......K..7.
005081EC   A9 1E AB 86  CA 3A 1E 12  B7 F4 9A 2A  C4 76 84 13  79 95 09 FA  .....:.....*.v..y...
00508200   AB D1 86 7C  53 0D BF 84  E9 B3 3F 42  13 EB 02 EE  A8 15 CC 79  ...|S.....?B.......y
00508214   20 01 5C 41  0F 21 2A 99  27 78 A6 E8  45 BB 67 A3  10 DE 45 5E   .\A.!*.'x..E.g...E^
00508228   8F 38 6B E2  5C 6E 44 09  86 E0 E6 B7  B3 77 67 80  21 04 BF 20  .8k.\nD......wg.!..
0050823C   C7 FC BD 9D  77 A5 4D 8C  C5 38 38 F6  8D 52 2B FC  56 43 D5 5A  ....w.M..88..R+.VC.Z
00508250   AE 2F AD B9  A2 51 D1 D0  53 45 03                    ./...Q..SE.
```

APEv2

3.1 APEv2

This is how information is laid out in an APEv2 tag:

APE Tags *Header*	32 bytes
APE Tag *Item* 1	10.. bytes
APE Tag *Item* 2	10.. bytes
...	10.. bytes
APE Tag *Item* n-1	10.. bytes
APE Tag *Item* n	10.. bytes
APE Tags **Footer_**	32 bytes

APE tag items should be sorted ascending by size. When streaming, parts of the APE tags can be dropped to reduce danger of drop outs between titles. This is not a must, but strongly recommended. Actually the items should be sorted by importance/byte, but this is not feasible. Only break this rule if you add less important small items and you don't want to rewrite the whole tag. An APE tag at the end of a file (strongly recommended) must have at least a footer, an APE tag in the beginning of a file (strongly unrecommended) must have at least a header. When located at the end of an MP3 file, an APE tag should be placed after the the last frame, just before the ID3v1 tag (if any).

3.1.1 APE Tags Header/Footer

Contains number, length and attributes of all tag items

Header and Footer are different in 1 bit in the Tags Flags to distinguish between them.

Member of APE Tag 2.0

Preamble	64 bits	{ ‘A’, ‘P’, ‘E’, ‘T’, ‘A’, ‘G’, ‘E’, ‘X’ }
Version Number, Bits 0...7 Version Number, Bits 8...15 Version Number, Bits 16...23 Version Number, Bits 24...31	32 bits	1000 = Version 1.000 (old) 2000 = Version 2.000 (new)
Tag Size, Bits 0... 7 Tag Size, Bits 8...15 Tag Size, Bits 16...23 Tag Size, Bits 24...31	32 bits	Tag size in bytes including footer and all tag items excluding the header to be as compatible as possible with APE Tags 1.000
Item Count, Bits 0... 7 Item Count, Bits 8...15 Item Count, Bits 16...23 Item Count, Bits 24...31	32 bits	Number of items in the Tag (n)
Tags Flags, Bits 0... 7 Tags Flags, Bits 8...15 Tags Flags, Bits 16...23 Tags Flags, Bits 24...31	32 bits	Global flags of all items (there are also private flags for every item)
Reserved	64 bits	Must be zero

3.1.2 Ape Tags Flags

Contains attribute of the tag (bit 31...) and of a item (bit 0...)

Member of APE Tags Header, Footer or Tag item

Note: APE Tags 1.0 do not use any of the APE Tag flags. All are set to zero on creation and ignored on reading.

Bit 31

- 0: Tag contains no header
- 1: Tag contains a header

Bit 30

- 0: Tag contains a footer
- 1: Tag contains no footer

Bit 29

- 0: This is the footer, not the header
- 1: This is the header, not the footer

Bit 28...3 Undefined, must be zero

Bit 2...1

- 0: Item contains text information coded in UTF-8
- 1: Item contains binary information*
- 2: Item is a locator of external stored information**
- 3: reserved

Bit 0

- 0: Tag or Item is Read/Write
- 1: Tag or Item is Read Only

[*] Binary information: Information which should not be edited by a text editor, because

- Information is not a text.
- Contains control characters
- Contains internal restrictions which can't be handled by a normal text editor
- Can't be easily interpreted by humans.

[**] Allowed formats:

- http://host/directory/filename.ext
- ftp://host/directory/filename.ext
- filename.ext
- /directory/filename.ext
- DRIVE:/directory/filename.ext

Note: Locators are also UTF-8 encoded. This can especially occur when filenames are encoded.

3.1.3 APE Tag Item

An APE tag item is a value assigned by a key.

Member of APE Tag Version 2.0

Note:

- APE Tags Item Key are case sensitive.
- Nevertheless it is forbidden to use APE Tags Item Key which only differs in case.
- And nevertheless Tag readers are recommended to be case insensitive.
- Every Tag Item Key can only occures (at most) once. It is not possible to transmit a Tag Key multiple time to change it contents.
- Tags can be partially or complete repeated in the streaming format.
- This is to make it possible to display artist and title if you missed the start of the transmission.
- It is recommended to transmit very important information like artist / album / title every 2 minutes and additional 5...10 seconds before the end. Be careful and don't transmit these information too often or during passages with high bitrate demand to avoid unnecessary drop-outs.

Size of the Item Value, Bits 0...7 Size of the Item Value, Bits 8...15 Size of the Item Value, Bits 16...23 Size of the Item Value, Bits 24...31	32 bits	Length len of the assigned value in bytes
Item Flags, Bits 0...7 Item Flags, Bits 8...15 Item Flags, Bits 16...23 Item Flags, Bits 24...31	32 bits	Item flags
Item Key	m bytes	Item key, can contain ASCII characters from 0x20 (Space) up to 0x7E (Tilde)
0x00	1byte	Item key terminator
Item Value	len bytes	Item value, can be binary data or UTF-8 string

3.1.4 APE Key

- An APE tag item key is a key for accessing special meta-information in an audio file.
- Member of APE Tag Item.
- APE tag item keys can have a length of 2 (including) up to 255 (including) characters in the range from 0x20 (Space) until 0x7E (Tilde).
- Typical keys should have a length of 2 ... 16 characters using the following characters: Space (0x20), Slash (0x2F), Digits (0x30...0x39), Letters (0x41...0x5A, 0x61...0x7A).
- Values can contain binary data, a value or a list of values. See here. List of values can be mixed, i.e. contain UTF-8 strings and external references beginning with file://..., http://www..., ftp://ftp....
- Not allowed are the following keys: ID3, TAG, OggS and MP+.

Currently the following keys are defined:

TODO

CHAPTER 4

MP4

- QuickTime File Format Specification
- ISO_IEC_14496-12
- ISO_IEC_14496-14

4.1 Multivalue Tags

Note: not finished..

The only case where iTunes writes multiple values is the `covr` atom by including multiple `data` atoms.

For a collection of 1700 random mp4s the only place where multiple values occur is multiple `----` atoms with `com.apple.iTunes:unknown` as identifier.

Foobar2000 (1.3.3)

Writing: Foobar writes single values in the official atoms and multiple values in the reverse DNS namespace. Multiple values get saved as multiple `---` with the same name and one *data* each.

Reading: Foobar only reads the first *data* atom in all child atoms in `ilst` and `----`. Everything else gets ignored and thrown out on save.

iTunes (11.4)

Writing: Given two `\xa9gen` atoms with each two `data` atoms. Modifying the genre modifies the first data atom of the second `\xa9gen` atom and it leaves the second `data` as is and writes is back. The first `\xa9gen` atom gets thrown out. If the genre gets removed in the iTunes GUI, both `\xa9gen` get thrown out.

Reading: Given two `\xa9gen` atoms with each two `data` atoms it displays the first `data` of the second `\xa9gen`.

No idea how it handles `---` atoms.

So, multiple values for official `ilst` sub atoms should be saved as multiple `data` because iTunes will at least not touch other values and editing/display use the first one.

4.2 Random Struct Decls

```
class DecoderConfigDescriptor extends BaseDescriptor :
        bit(8) tag=DecoderConfigDescrTag {
    bit(8) objectTypeIndication;
    bit(6) streamType;
    bit(1) upStream;
    const bit(1) reserved=1;
    bit(24) bufferSizeDB;
    bit(32) maxBitrate;
    bit(32) avgBitrate;
    DecoderSpecificInfo decSpecificInfo[0 .. 1];
    profileLevelIndicationIndexDescriptor profileLevelIndicationIndexDescr[0..255];
}
```

```
class ES_Descriptor extends BaseDescriptor :
        bit(8) tag=ES_DescrTag {
    bit(16) ES_ID;
    bit(1) streamDependenceFlag;
    bit(1) URL_Flag;
    bit(1) OCRstreamFlag;
    bit(5) streamPriority;
    if (streamDependenceFlag)
        bit(16) dependsOn_ES_ID;
    if (URL_Flag) {
        bit(8) URLlength;
        bit(8) URLstring[URLlength];
    }
    if (OCRstreamFlag)
        bit(16) OCR_ES_Id;
    DecoderConfigDescriptor decConfigDescr;
    if (ODProfileLevelIndication==0x01) //no SL extension.
    {
        SLConfigDescriptor slConfigDescr;
    }
    else // SL extension is possible.
    {
        SLConfigDescriptor slConfigDescr;
    }
    IPI_DescrPointer ipiPtr[0 .. 1];
    IP_IdentificationDataSet ipIDS[0 .. 255];
    IPMP_DescriptorPointer ipmpDescrPtr[0 .. 255];
    LanguageDescriptor langDescr[0 .. 255];
    QoS_Descriptor qosDescr[0 .. 1];
    RegistrationDescriptor regDescr[0 .. 1];
    ExtensionDescriptor extDescr[0 .. 255];
}
```

```
abstract class DecoderSpecificInfo extends BaseDescriptor :
    bit(8) tag=DecSpecificInfoTag
{
    // empty. To be filled by classes extending this class.
}
```

AudioSpecificConfig if objectTypeIndication == 0x40 and streamType = 0x5

```
AudioSpecificConfig extends DecoderSpecificInfo
{
```

```
    bit(5) audioObjectType;
    if (audioObjectType == 31) {
        audioObjectType = 32 + bit(6) audioObjectTypeExt;
    }

    [...]
}
```

audioObjectType is defined in 14496-3 1.6.2.2.1

CHAPTER 5

ASF

5.1 Multiple Values

(Tested with Windows 8)

The following text keys support multiple values in Win8:

Author, WM/Composer, WM/Conductor, WM/Producer, WM/Category, WM/Genre

For saving/reading multiple values:

- For tags which can be in ContentDescription (Author) write the first value there, the rest in MetadataLibrary.
- For tags which can be in ExtendedContentDescription (WM/Composer) write the first value there and the rest in MetadataLibrary.
- When reading, the order in which the objects appear in the file doesn't matter. First value from ContentDescription, rest from MetadataLibrary etc.

CHAPTER 6

Ogg

- Xiph Formats
  - Ogg bitstream structure
  - Vorbis comment structure
  - Ogg Vorbis embedding
  - FLAC format, and Ogg FLAC embedding
  - Ogg Theora embedding