Mp3 Info Tag rev 1 specifications - draft 0

The purpose of this tag is to provide extra information about the mp3 bistream, encoder and parameters used. This tag should, as much as possible, be meaningfull for as many encoders as possible, even if it is unlikely that other encoders than Lame will implement it.

This tag should be backward compatible with tha Xing vbr tag, providing basic support for a lot of already written software. As much as possible the current revision (revision 1) should provide information similar to the one already provided by revision 0.

A few fields, as they could be necessary for some functionnalities of already existing software, should not be moved in any version of the tag. They are indicated as "UNMOVABLE".

LAME 3.88 Tag example :

frame at 44.1kHz samplerate:

```
       ?Éd

        Xing   ?   t
0000: FF FB 90 64-00 00 00 00-00 00 00 00-00 00 00 00  0? ?•????¶????
0010: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00  ?#&(*,.3579<>@DG
0020: 00 00 00 00-58 69 6E 67-00 00 00 0F-00 00 00 74  IKMOTVXZ]_cfhjln
0030: 00 00 30 C1-00 04 07 09-0B 0D 0F 14-16 18 1A 1D  suwy|~ÇäçëïîÅöûÿ
0040: 1F 23 26 28-2A 2C 2E 33-35 37 39 3C-3E 40 44 47  Ü¥ƒúª¿¬¼«???????
0050: 49 4B 4D 4F-54 56 58 5A-5D 5F 63 66-68 6A 6C 6E  ????????????µ??
0060: 73 75 77 79-7C 7E 80 84-87 89 8B 8D-8F 94 96 98  ????????  XLAME
0070: 9A 9D 9F A3-A6 A8 AA AC-AE B3 B5 B7-B9 BC BE C0  3.88 (beta)
0080: C4 C7 C9 CB-CD CF D4 D6-D8 DA DD DF-E3 E6 E8 EA
0090: EC EE F3 F5-F7 F9 FC FE-00 00 00 58-4C 41 4D 45
00A0: 33 2E 38 38-20 28 62 65-74 61 29 00-00 00 00 00
00B0: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00C0: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00D0: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00E0: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00F0: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
0100: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
0110: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
0120: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
0130: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
0140: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
0150: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
0160: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
0170: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
0180: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
0190: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
01A0: 00
```

/*

```
//   ZONE A - Traditional Xing VBR Tag data
//    4 bytes for Header Tag
//    4 bytes for Header Flags
//  100 bytes for entry (NUMTOCENTRIES)
//    4 bytes for FRAME SIZE
//    4 bytes for STREAM_SIZE
//    4 bytes for VBR SCALE. a VBR quality indicator: 0=best 100=worst

//   ZONE B - Initial LAME info
//   20 bytes for LAME tag.  for example, "LAME3.12 (beta 6)"
// ____________
//  140 bytes
//

//   ZONE C - LAME Tag
//   208 bytes unused in 128k frame (in 48kHz case)
//
//   using
//   FrameLengthInBytes = 144 * BitRate / SampleRate + Padding
//
//   this gives
//   Layer III, BitRate=128000, SampleRate=44100, Padding=0
//        ==>  FrameSize=417 bytes
//   Layer III, BitRate=128000, SampleRate=48000, Padding=0
//        ==>  FrameSize=384 bytes
//
//   so this would make the minimal frame size 384 bytes ($0-$17F), hence the available bytes for this field are not 241 as in this 44100Hz case, but at most 208 bytes.
*/
```

frame at a 48.0kHz samplerate:

```
0000: FF FB 94 64-00 00 00 00-00 00 00 00-00 00 00 00  ?öd
0010: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
0020: 00 00 00 00-58 69 6E 67-00 00 00 0F-00 00 00 7E   Xing  ?  ~
0030: 00 00 30 C0-00 04 06 08-0C 0E 10 12-16 18 1A 1C   0? ???????????
0040: 21 23 25 27-2B 2D 2F 31-35 37 39 3B-3F 41 43 47  !#%'+-/1579;?ACG
0050: 49 4B 4D 51-53 55 57 5B-5D 5F 62 66-68 6A 6C 70  IKMQSUW[]_bfhjlp
0060: 72 74 76 7A-7C 7E 80 84-86 88 8C 8E-90 92 96 98  rtvz|~Çäåêî?ÉÆûÿ
0070: 9A 9C A1 A3-A5 A7 AB AD-AF B1 B5 B7-B9 BB BF C1  Ü£íúÑº½¡»???????
0080: C3 C7 C9 CB-CD D1 D3 D5-D7 DB DD DF-E2 E6 E8 EA  ????????????µ??
0090: EC F0 F2 F4-F6 FA FC FE-00 00 00 58-4C 41 4D 45  ????÷·??   XLAME
00A0: 33 2E 38 38-20 28 62 65-74 61 29 00-00 00 00 00  3.88 (beta)
00B0: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00C0: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00D0: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00E0: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00F0: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
0100: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
0110: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
0120: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
0130: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
0140: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
0150: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
0160: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
0170: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
```

Suggested Info Tag extension fields + layout :

REMARK:

In the Info Tag, the "**Xing**" identification string (mostly at 0x24) of the header is replaced by "**Info**" in case of a CBR file. This was done to avoid CBR files to be recognized as traditional Xing VBR files by some decoders. Although the two identification strings "Xing" and "Info" are both valid, it is suggested that you keep the identification string "Xing" in case of VBR bistream in order to keep compatibility.

now:

LAME VBR & ABR:

"Xing"

LAME CBR:

"Info"

								00h	00h	00h	$9B	"L"	"A"	"M"	"E"
$A0	"."	$A2	$A3	$A4	$A5	$A6	$A7	$A8	$A9	$AA	$AB	$AC	$AD	$AE	$AF
$B0	$B1	$B2	$B3	$B4	$B5	$B6	$B7	$B8	$B9	$BA	$BB	$BC	$BD	$BE	$BF

byte $9B	VBR Quality

This field is there to indicate a quality level, although the scale was not precised in the original Xing specifications.

In case of Lame, the meaning is the following:

int Quality = (100 - 10 * gfp->VBR_q - gfp->quality)h

examples:

V0 and q0 = 100 - 10 * 0 - 0 = 100 => 64h
V0 and q2 = 100 - 10 * 0 - 2 = 98 => 62h
V2 and q5 = 100 - 10 * 2 - 5 = 75 => 4Bh
V9 and q9 = 100 - 10 * 9 - 9 = 1 => 01h

bytes $9A-$A4	Encoder short VersionString

9 characters

examples:

"LAME3.90a" : LAME version 3.90 alpha
"GOGO3.02b" : GOGO version 3.02 beta

byte $A5	Info Tag revision + VBR method

two 4 bits fields:

- 4 MSB: Info Tag revision, range 0d-15d
  Possible values:
  0: rev0
  1: rev1
  15: reserved

- 4 LSB: VBR Method, range 0d-15d
  Indicate the VBR method used for encoding.
  Possible values:

0d	unknown
1d	constant bitrate
2d	restricted VBR targetting a given average bitrate (ABR)
3d	full VBR method1
4d	full VBR method2
5d	full VBR method3
6d	full VBR method4
7d	
8d	constant bitrate 2 pass
9d	abr 2 pass
10d	
11d	
12d	
13d	
14d	
15d	reserved

In case of Lame, the meaning is the following:
2: abr
3: vbr old / vbr rh
4: vbr mtrh
5: vbr mt

examples:

byte $A5 = 03h

= 0001 0011b =>

- 4 MSB = 0001b =1d : LAME Tag revision 1
- 4 LSB = 0011b = 3d : vbr-old / vbr-rh

byte $A5 = 35h

= 0011 0101b =>

- 4 MSB = 0011b = 3d : LAME Tag revision 3
- 4 LSB = 0101b = 5d : vbr-new / vbr-mt

---

byte $A6	Lowpass filter value

---

int lowpass = (lowpass value) / 100

range: 01h = 01d : 100Hz -> FFh = 255d : 25500Hz

value 00h => unknown

examples:

byte $A6 = C3h

C3h = 195d : 19500Hz

byte $A6 = 78h

78h = 120d : 12000Hz

---

bytes $A7-$AF	Replay Gain

---

as defined here: http://www.david.robinson.org/replaylevel/ by David Robinson

three fields:

- bytes $A7-$AA : 32 bit floating point **"Peak signal amplitude"** 32 bit floating point. 00h00h00h00h : unknown

  1.0 is maximal signal amplitude storeable in decoding format.
  0.8 is 80% of maximal signal amplitude storeable in decoding format.
  1.5 is 150% of maximal signal amplitude storeable in decoding format.

  info + examples: will follow

- byte $AB-$AC : 16 bit **"Radio Replay Gain"** field, required to make all tracks equal loudness.

  from http://privatewww.essex.ac.uk/~djmrob/replaygain/rg_data_format.html

  bits 0h-2h: NAME of Gain adjustment:
  000 = not set
  001 = radio
  010 = audiophile
  (see - room for plenty more!)

bits 3h-5h: ORIGINATOR of Gain adjustment:
000 = not set
001 = set by artist
010 = set by user
011 = set by my model
100 = set by simple RMS average
etc etc (see - room for plenty more again!)

bit 6h: Sign bit
bits 7h-Fh: ABSOLUTE GAIN ADJUSTMENT
storing 10x the adjustment (to give the extra decimal place).

- byte $AD-AE : 16 bit **"Audiophile Replay Gain"** field, required to give ideal listening loudness

from http://privatewww.essex.ac.uk/~djmrob/replaygain/rg_data_format.html

bits 0h-2h: NAME of Gain adjustment:
000 = not set
001 = radio
010 = audiophile
(see - room for plenty more!)

bits 3h-5h: ORIGINATOR of Gain adjustment:
000 = not set
001 = set by artist
010 = set by user
011 = set by my model
100 = set by simple RMS average
etc etc (see - room for plenty more again!)

bit 6h: Sign bit
bits 7h-Fh: ABSOLUTE GAIN ADJUSTMENT
storing 10x the adjustment (to give the extra decimal place).

---

byte $AF	Encoding flags + ATH Type

---

two 4 bits fields:

- 4 MSB: 4 encoding flags

000?	LAME uses "--nspsytune", ? = 0 : false ? = 1 : true
00?0	LAME uses "--nssafejoint" ? = 0 : false ? = 1 : true
0?00	This track is --nogap continued in a next track ? = 0 : false ? = 1 : true is true for all but the **last** track in a --nogap album
?000	This track is the --nogap continuation of an earlier one ? = 0 : false ? = 1 : true

	is true for all but the **first** track in a --nogap album

- 4 LSB: LAME ATH Type, range 0d-15d

examples:

byte $AF = 03h

= 0000 0011b =>

- 4 MSB = 0000b = 0d : LAME does NOT use --nspsytune, LAME does NOT use --nssafejoint
- 4 LSB = 0011b = 3d : ATH type 3 used

byte $AF = 15h

= 0001 0101b =>

- 4 MSB = 0001b = 3d : LAME does use --nspsytune, LAME does NOT use --nssafejoint
- 4 LSB = 0101b = 5d : ATH type 5 used

---

byte $B0	if ABR {specified bitrate} else {minimal bitrate}

IF the file is an ABR file:

range: 01h = 01d : 1 kbit/s (--abr 1) -> FFh = 255d : 255 kbit/s or larger (--abr 255)

value 00h => unknown

examples:

byte $B0 = C3h

C3h = 195d : --abr 195

byte $B0 = 78h

78h = 128d : --abr 128

byte $B0 = FEh

FEh = 254d : --abr 254

byte $B0 = FFh

FEh = 255d : --abr 255 or higher, eg: --abr 280

IF the file is NOT an ABR file: (CBR/VBR)

the (CBR)/(minimal VBR (-b)) bitrate is stored here 8-255. 255 if bigger.

examples:

- "LAME -V0 -b224" will store (224)d=(E0)h
- "LAME -b320" will store (255)d=(FF)h

---

bytes $B1-$B3 Encoder delays

store in 3 bytes:

[xxxxxxxx][xxxxyyyy][yyyyyyyy]

the 12 bit values (0-4095) of how many samples were added at start (encoder delay) in X and how many 0-samples were padded at the end in Y to complete the last frame.

so ideally you could do: #frames*(#samples/frame)-(these two values) = exact number of samples in original wav.

so worst case scenario you'd have a 48kHz file which would give it a range of 0.085s at the end and at the start.

example:
[01101100][00010010][11010010]

X = (011011000001)b = (1729)d, so 1729 samples is the encoder delay
Y = (001011010010)b = (722)d, so 722 samples have been padded at the end of the file

byte $B4 Misc

2 lsb	I'd like to add the different noise shapings also in a 2-bit field (0-3) (00)b: noise shaping: 0 (01)b: noise shaping: 1 (10)b: noise shaping: 2 (11)b: noise shaping: 3
3 bits	Stereo mode msb fist: (000)b: (m)ono (001)b: (s)tereo (010)b: (d)ual (011)b: (j)oint (100)b: (f)orce (101)b: (a)uto (110)b: (i)ntensity (111)b: (x)undefined / different
1 bit	unwise settings used (0)b: no (1)b: yes (definition encoder side(*))
2 msb	**Source** (not mp3) sample frequency (00)b: 32kHz or smaller (01)b: 44.1kHz (10)b: 48kHz (11)b: higher than 48kHz

(*)some settings were used which would likely damage quality in normal circumstances. (like disabling all use of the ATH or forcing only short blocks, -b192 ...)

byte $B5 MP3 Gain

any mp3 can be amplified by a factor **2 ^ (x * 0.25)** in a lossless manner by a tool like eg. mp3gain

byte $B5 is set to (00)h by default.

if done so, this 8-bit field can be used to log such transformation happened so that any given time it can be undone.

WARNING:

Do **NOT** alter this field if you do not fully understand its use. You will damage the Replaygain fields and musicCRC if you do not implement this correctly.

You can **only** modify this field if

1. the TagCRC checks out
2. you update **all three** the ReplayGain fields with the correct number of 1.5dB steps
3. the TagCRC is updated after all this

Do NOT change/update the musicCRC. It will be invalid after you change this mp3gain field.
If an application like mp3gain changes the main music frames of the mp3 then the musicCRC should be invalid.
The Lame Tag CRC should still be valid however (it could be updated by mp3gain).

only tools like mp3gain should use this field, as it is made for making lossless adjustments to the mp3 after encoding is finished. No need to support this in LAME or any decoder at all.

2^(a/4) , range of "a" here: -127..0..127

[byte $B5]b	a	dB change	amplification factor used was
[11111111]	-127	-190.5dB	0.000000000276883
...	...		
[10000011]	-3	-4.5dB	0.594603557501360533
[10000010]	-2	-3dB	0.707106781186547524
[10000001]	-1	-1.5dB	0.840896415253714543
[00000000]	0	0dB	1.0
[00000001]	1	+1.5dB	1.18920711500272107
[00000010]	2	+3dB	1.41421356237309505
[00000011]	3	+4.5dB	1.68179283050742909
[00000100]	4	+6dB	2.0
[00000101]	5	+7.5dB	2.37841423000544213
...	...		
[00011111]	31	+46.5dB	215.269482304950923
...	...		
[01111111]	127	+190.5dB	3611622602.83833951

the +-190.5dB range is too large, but there was little else to do with the extra bits, so for uniformity we took this range.

bytes $B6-$B7 | Preset and surround info

2 most significant bits: unused

3 bits: surround info

0: no surround info
1: DPL encoding
2: DPL2 encoding
3: Ambisonic encoding
8: reserved

11 least significant bits: Preset used.

0: unknown/ no preset used
This allows a range of 2047 presets. With Lame we would use the value of the internal preset enum.

---

bytes $B8-$BB	MusicLength

---

32 bit integer filed containing the exact length in bytes of the mp3 file originally made by LAME excluded ID3 tag info at the end.

The first byte it counts is the **first byte of this LAME Tag** and the last byte it counts is the **last byte of the last mp3 frame containing music**.

Should be filelength at the time of LAME encoding, except when using ID3 tags.

practical example:
~~**[misc+ID3v2 tag info]**~~**[LAME Tag frame][complete mp3 music data]**~~**[misc+ID3v1/2 tag info]**~~

remark: applying **any** (ID3v2) kind of tagging or information in FRONT of the LAME/Xing Tag frame is a very bad idea. You will disable the functionality of all decoders to read the tag info correctly. (for example: VBR mp3 seek info will no longer be usable)

range (1)d-(4,294,967,295)d [or about 4294967295/(650*1024*1024)/320*1411 = 27.79 hours of 44.1kHz 320kbit/s music.]

Musiclength not set / unknown / larger than 4G:

$B8	$B9	$BA	$BB
00h	00h	00h	00h

use of this field: together with the next field deliver

- an effective verification of music's integrity
- and effective shield from TAGs.

Examples:

$B8	$B9	$BA	$BB
(29)h	(17)h	(A3)h	(62)h

would be (2917A362)h = (689,415,010)b bytes

$B8	$B9	$BA	$BB
(00)h	(3B)h	(82)h	(B5)h

would be (3B82B5)h = (3,900,085)b bytes

---

bytes $BC-$BD | MusicCRC

---

contains a CRC-16 of the complete mp3 music data as made originally by LAME.

- with as first byte the **first byte of the first mp3 frame containing music**. (the frame right after this LAME Tag)
- with as last byte the **last byte of the last mp3 frame containing music**. (so that additional tags at the end are not included.)

reason : will guarantee that the actual mp3 music data is intact, unregardless people adding ID3 tags to the end (or the start) of the file.

practical example:
~~**[misc+ID3v2 tag info]**~~**[LAME Tag frame]**[complete mp3 music data as made by LAME]~~**[misc+ID3v1/2 tag info]**~~

remark: applying **any** (ID3v2) kind of tagging or information in FRONT of the LAME/Xing Tag frame is a very bad idea. You will disable the functionality of all decoders to read the tag info correctly. (for example: VBR mp3 seek info will no longer be usable)

Meaning of this musicCRC:

> "if the musicCRC is correct, then this file (or the music data in it) are identical to when encoded by LAME"

It does not say:

- this is a valid mp3
- this mp3 sounds good
- this mp3 has been downloaded correctly off the internet
- it has not been retagged (very possible)

This will enable:

- effective verification method to see if the *music data* is exactly what it was at time of encoding
- swift and accurate removal of any sort of ID3 or different tags
  1) search for start of a LAME Tag (pos $xy)
  2) if CRC($(xy+c0)..$(xy+MusicLength)) corrent then cut out $(xy+c0)..$(xy+MusicLength)

CRC-16 should suffice since in the event of total randomness every 65536th **defective** file on average will be falsely identified as being not defective. Also, CRC-16 routines are present in both mp3 encoder and decoders.

CRCInitValue := $0000;

---

bytes $BE-$BF | CRC-16 of Info Tag

---

contains a CRC-16 of the first 190 bytes ($00-$BD) of the Info header frame. This field is calculated at the end, once all other fields are completed.

reason : safeguards LAME VBR header against easy tampering. Improving the header functionality as quality control/ verification tool for VBR files.

CRCInitValue := $0000;

Remarks / Ideas:

- The idea behind storing all this specific (quality significant) LAME settings used to encode in the mp3 itself is to give some kind of quality assurance for (mainly) non-CBR files. You would download an mp3 from the web and know exactly what kind of setting was used on this file. Now there is no way to keep find out if your download is a "-V9 --lowpass 12" or a "-V1 --lowpass 19.5" file. Even though both have a significant quality difference.
- These headers should also be placed in front of **CBR** files. Given the Bitrate is >=64 the frame could be housed in a CBR file using the same framesize as the mp3 itself.
- If limiting the LAME string to 9 bytes "LAME X.YZu", the extension revision 0 could take 27 bytes and it would still fit a 64 kbit 48kHz frame:

```
0000: FF FB 54 04-00 00 00 00-00 00 00 00-00 00 00 00  ?T?
0010: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
0020: 00 00 00 00-58 69 6E 67-00 00 00 0F-00 00 00 7E    Xing   ?   ~
0030: 00 00 2F E1-00 04 04 04-04 04 04 05-09 0B 0D 0F    /ß ??????????
0040: 13 15 17 19-1D 1F 21 23-27 29 2B 2D-31 33 35 39  ?§????!#')+-1359
0050: 3B 3D 3F 43-45 47 49 4D-4F 51 53 57-59 5B 5D 61  ;=?CEGIMOQSWY[]a
0060: 63 65 67 6B-6D 6F 73 75-77 79 7D 7F-81 83 87 89  cegkmosuwy}?üâçë
0070: 8B 8D 91 93-95 97 9B 9D-9F A1 A5 A7-A9 AB AF B1  ïìæôòù¢¥ƒíÑº?½»?
0080: B3 B7 B9 BB-BD C1 C3 C5-C7 CB CD CF-D1 D5 D7 D9  ????????????????
0090: DB DF E1 E3-E5 E9 EB ED-00 00 00 0B-4C 41 4D 45  ??ß?????   ?LAME
00A0: 33 2E 37 30-00 00 00 00-00 00 00 00-00 00 00 00  3.70
00B0: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
```

- In these last two cases, instead of the now default 128kbit/s one, in case of non-CBR, LAME could add any (64 .. 320) sized tag to the mp3, being as close as possible to the average bitrate of the file. So a VBR file averaging 172 kbit would get a 160kbit/s LAME Tag, an ABR file averaging 253 kbit/s a 256kbit/s LAME Tag... (useful for non-vbr compliant decoders guessing file length based on size first frame)