

.herb File Format Documentation

by herb delux

Version 1.0



Introduction:

This document serves as a breakdown of the header structure and frame writing structure of a .herb audio file.

.herb audio files are lossless uncompressed PCM (pulse code modulation) encoded audio files, similar to a .wav file, written with all header and frames values using big endian. Header items are unsigned integers (when applicable) and frames are signed. .herb files also have a different header structure, and support metadata (track info) tags, as you can see below.

Because the header length is written into the header itself, even if you write a program that has problems processing the metadata (track info) or finding the final b'herb', you can still read the important information (header length, bit depth, number of channels, and sample rate) so that you can at least find and play the audio frames.

Header structure:

Header is a minimum of 50 bytes without metadata (track info). With metadata it is variable in length.

4 bytes: b'herb'

A byte string of 'herb' in lowercase. This symbolizes that the file is a .herb file and it's header is beginning.

4 bytes: b'pcmb'

This byte string represents that the file uses PCM (Pulse code modulation) for the audio with big endian for its header items and audio frames. This is intended to never change. It also represents the beginning of the main and most important part of the header: the information about bit depth, sample rate, number of channels, etc.

4 bytes: header length

An integer, unsigned, in big endian, representing the total length of the header for this file. This allows you, after reading the sample rate, bit depth, and number of channels, to know exactly where the audio frames start (in case of metadata corruption or issues, etc).

3 bytes: number of channels

An integer, unsigned, in big endian, for the number of audio channels embedded in the file. 16,777,215 is thus the max number of channels supported.

2 byte: bit depth

An integer, unsigned, in big endian, for the bit depth of the file. 65,535 is thus the max number of bits per sample supported.

4 bytes: Sample rate

An integer, unsigned, in big endian, for the sample rate of the file. 4,294,967,295 is thus the max sample rate supported.

10 bytes: full size of the audio data (audio frames), not including the header.

An integer, unsigned, in big endian, representing the size in bytes of the audio data portion of the file. If you add this with the header length, in theory, you should get the full size of the .herb file. Thus, this value plus the header length in bytes should also give you the full file size in bytes.

10 bytes: empty space of 0's for future additions

Written as an integer, unsigned, in big endian, not that it matters at the moment.

4 bytes: b'tags'

Signifies the potential beginning of the metadata section (track info). If the next byte (number of metadata items) is zero, then there is no metadata. If the value (of the next byte) is 1 or more, then there are metadata items.

1 byte: number of metadata items

An integer, unsigned, in big endian, representing how many metadata items or entries there are coming up in the header. 255 items max.

IF THERE IS NO METADATA (the previous 1 byte was the integer 0):**4 bytes: b'leaf'**

This byte string represents the end of the header, and the beginning of the audio frames.

IF THERE IS METADATA (the previous 1 byte was an integer greater than 0):**4 bytes: Byte string of metadata (pepper) section name.**

Examples: b'trck', b'arti', b'albm', b'year', etc

1 byte: length of the actual metadata information (the track name itself, the artists name, etc)

An integer, unsigned, in big endian. This represents the length in bytes of the actual important part of the metadata.

This value is allowed to be zero.

N bytes: the actual meta data. (n = the integer from the 2 bytes prior, if not zero)

Examples: “The Official Sound Track” or “Blues Singer” etc.

This section is n number of bytes long, with n being the integer gathered while reading the prior two bytes.

4 bytes: Byte string of metadata section name (next item of metadata)

Examples: b'trck', b'arti', b'albm', b'year', etc

1 byte: length of the actual metadata information (the track name itself, the artists name, etc)

An integer, unsigned, in big endian. This represents the length in bytes of the actual important part of the metadata.

This value is allowed to be zero.

N bytes: the actual meta data. (n = the integer from the 2 bytes prior, if not zero)

Examples: “The Official Sound Track” or “Blues Singer” etc.

This section is n number of bytes long, with n being the integer gathered while reading the prior two bytes.

Then keep repeating for all metadata sections you want to add.

***Note:** The herb sauce program adds in up to 5 optional meta data tag sections: track name, artist name, album name, year released, and track number.*

However, you don't have to follow this, as long as you follow the structure laid out for a .herb file itself in this document. You can have up to 255 metadata items, including custom ones you make yourself. For example b'vart' could be a section for “Various Artists” or something like that. See the end of this section for recommended metadata tags.

Note: You can calculate the length of the metadata tags section in bytes, by taking the header length value and subtracting 50 (since the bare minimum of header data is 50 bytes)

4 bytes: b'leaf'

This byte string represents the end of the header, and in this case the metadata section too, and the beginning of the audio frames. Remember, there is only one b'leaf'. Its location in the header depends on how many metadata items there are or aren't.

Metadata tags:

Here are the recommended (but not required) metadata tags in the recommended (but not required) order.

b'trck' – Track name

b'arti' – Artist name

b'albm' – Album name

b'year' – Year released

b'tnum' – Track number

b'ofnm' – OF how many tracks total

b'comp' – Composer(s)

b'prod' – Producer(s)

b'feat' – Featuring artists

b'imag' – File name of PNG or JPG album art. Must exist in the same directory as the .herb file only

b'spos' – Speaker layout or positioning (hopefully will add to this down the line)

b'mixx' – Mixer engineer(s)

b'mast' – Mastering engineer(s)

b'gear' – Tools, software, or gear used

b'note' – Custom notes

b'not2' – Custom note 2

Feel free to create your own. You technically don't have to follow that order either, it's just recommended. You only need to follow the header structure laid out above (4 byte bytes string, 2 bytes for length of actual data, then actual data), and the frame writing/reading structure (reminder, .herb files are PCM encoded audio files).

For reading and parsing these tags I recommend using a dictionary as such (below is some pseudo Python code):

```
#b'tags'
```

```
bytetags = track.read(4)
```

```
#number of metadata items
```

```
tagnums = track.read(1)
```

```
tagnums = int.from_bytes(tagnums, 'big', signed=False)
```

```
tagdict = {
```

```
    b'trck' : " ",
```

```
    b'albm' : " ",
```

```
}
```

```
for i in range(0, tagnums):
```

```
    tagval = track.read(4)
```

```
    taglength = track.read(1)
```

```
    taglength = int.from_bytes(taglength, 'big', signed=False)
```

```
    data = track.read(taglength)
```

```

data = data.decode("utf-8")

if tagval in tagdict.keys():
    tagict[pepperval] = data
else:
    pass

trackname = tagdict[b'trck']
artistname = tagdict[b'arti']
#continue as needed for the track info you want to parse or store

```

Audio Frames:

Audio frames are written similarly to a .wav file, except in a .herb file they are written in big endian.
(Big endian, signed integers)

For a 2 channel .herb file at a bit depth of 24 (3 bytes), the frames are written: 3 bytes for sample #1 channel #1. Followed by 3 bytes for sample #1 channel #2. Then 3 bytes for sample #2 channel #1, followed by 3 bytes for sample #2 channel #2, and so on with that same pattern.

For a 4 channel .herb file at a bit depth of 24 (3 bytes), the frames are written: 3 bytes for sample #1 channel #1, 3 bytes for sample #1 channel #2, 3 bytes for sample #1 channel #3, 3 bytes for sample #1 channel number 4. Then 3 bytes for sample #2 channel #1, 3 bytes for sample #2 channel #2, 3 bytes for sample #2 channel #3, 3 bytes for sample #2 channel #4, and so on with that same pattern.

The same pattern of frame writing is used for .herb files regardless of how many channels there are in the file.

Short point about the “why” of this program and file type:

I wanted to make 4.0 music (generally front facing (dual stereo), but not required) easily accessible, and to also support metadata (track info). With another stereo bus to mix with, you can split up your tracks/instruments. For example, now your kick can come out the 1,2 channels, and the bass out of the 3,4. This means those low end frequencies aren't competing anymore electrically (or digitally). You end up with way more dynamics/dynamic range, texture, soundstage and original fidelity. Instead of 8 tracks going into a 2 channel stereo output, you have 4 tracks into a stereo output, and another 4 into the other. You can also ride your limiters and compressors less, while still getting the loudness you want. Everything (processing, amps, speakers, etc) should hopefully be more in the optimal zone for performance and quality.

(See how to support me and other license info on the next page!)

Other:

You can support me by purchasing my original music from Bandcamp or Deluxe herbs:

<https://deluxeherbs.net/artists/herbdelux/herbdelux.html>

<https://herbdeluxmusic.bandcamp.com>

For more information check out the GitHub page for my free open source .herb file player and creator called herb sauce at:

<https://github.com/herbdelux/herbsauce>

License:

The program herb sauce and the .herb lossless audio file format are released into the world, with the intention of being free and open source. Hopefully they are used for good purposes and not bad ones.

For further information regarding the licenses of the packages I import in my code, please check with their respective documentation, licenses, or developers.

The butter and peppers logo is a real picture I took with my phone of real peppers that I grew, with a real stick of butter that I bought, on a cheeseboard I have. I am making it public domain, since I use it throughout the open source herb sauce software.

This is not intended for use to infringe on other entities copyrights or trademark rights.

herb sauce and the .herb file type comes as is with no warranty

herb