BWF

— a format for audio data files in broadcasting

Supplement 3 — Peak Envelope Chunk

July 2001

EBU Broadcast
Wave Format

European Broadcasting Union
Case postale 45
Ancienne Route 17A
CH-1218 Grand-Saconnex
Geneva, Switzerland
techreview@ebu.ch

EBU·UER

Summary

This document specifies a new chunk for use with Broadcast Wave Format (BWF) files, to contain information about peak audio signal levels in the file. When audio files are exchanged between workstations, it can speed up the opening, display and processing of a file if such data is available before the audio data itself is read.

The addition of a <**levl**> chunk to a BWF file provides a standard for storing and transferring data about the signal peaks obtained by sub-sampling the audio. This data in the chunk can be used to provide the envelope of the audio essence in the file.

In addition, it is possible to send the peak-of-peaks, which is the first audio sample whose absolute value is the maximum value of the entire audio file. An audio application can use this information to normalize a file in real-time, without having to scan the entire file (since this has already been done by the sender).

EBU Broadcast
Wave Format

July 2001

Contents

EBU Broadcast
Wave Format

1. Introduction

When audio files are exchanged between workstations, it can speed up the opening, display and processing of a file if data is available about the peak audio signal levels in the file. The addition of a ***<levl>*** chunk to a Broadcast Wave Format (BWF) file [1] provides a standard for storing and transferring data about the signal peaks obtained by sub-sampling the audio. This data in the chunk can be used to provide the envelope of the audio essence in the file. This will allow an audio application to display the audio files quickly, without loosing too much accuracy.

In addition, it is possible to send the peak-of-peaks, which is the *first* audio sample whose absolute value is the maximum value of the entire audio file. An audio application can use this information to normalize a file in real-time without having to scan the entire file. (Since this has already been done by the sender).

1.1. Terminology

The audio signal is divided into blocks. One **peak frame** is generated for each audio block there. There are n **peak values** for each peak frame, where n is the number of peak channels. Each peak value may consist of one (positive only) or two (one positive and one negative) **peak points**.

1.2. Generation of peak values

The audio signal is divided into blocks of samples of constant size. The default, and recommended, size of the blocks is 256 samples from each channel.

The samples of each channel are evaluated to find the peak points (maximum values). It is recommended that separate peak points are found for positive and negative samples but alternatively only the absolute value (either positive or negative) may be used. All the peak points are unsigned values.

The peak points are rounded to one of two formats, either 8 or 16 bits. In most cases the 8-bit format is sufficient. The 16-bit format should cover any cases needing higher precision.

The formatted peak points for each channel are assembled into peak frames. Each peak frame contains the positve and negative peak points (or the absolute peak point) for each channel in the same order as the audio samples.

These peak frames are carried as the data in the Peak Envelope chunk. The peak envelope chunk starts with a header that contains information that allows the peak data to be interpreted.

The **peak-of-peaks** is the *first* audio sample whose absolute value is the maximum value of the entire audio file. Rather than storing the peak-of-peaks as a sample value, the *position* of the peak-of-peaks is stored. In other words, an audio sample frame index is stored. An application then knows *where* to read the peak-of-peaks in the audio file. It would be more difficult to store a value for peak since this is dependant on the binary format of the audio samples (integers, floats, double...).

2. Peak envelope chunk

2.1. Definition

The peak envelope, ***<levl>***, chunk consists of a header followed by the data of the peak points. The overall length of the chunk will be variable, depending on the audio content, the block size and how the peak data is formatted.

```
typedef struct peak_envelope
{
    CHAR        ckID[4];                // {'l','e','v','l'}
    DWORD       ckSize;                 // size of chunk
    DWORD       dwVersion;              // version information
    DWORD       dwFormat;               // format of a peak point
                                           1 = unsigned char
                                           2 = unsigned short
    DWORD       dwPointsPerValue;       // 1 = only positive peak point
                                           2 = positive AND negative peak point
    DWORD       dwBlockSize;            // frames per value
    DWORD       dwPeakChannels;         // number of channels
    DWORD       dwNumPeakFrames;        // number of peak frames
    DWORD       dwPosPeakOfPeaks;       // audio sample frame index
                                        // or 0xFFFFFFFF if unknown
    DWORD       dwOffsetToPeaks;        // should usually be equal to the size of this header, but
                                           could also be higher
    CHAR        strTimestamp[28];       // ASCII: time stamp of the peak data
    CHAR        reserved[60];           // reserved set to 0x00
    CHAR        peak_envelope_data[]    // the peak point data
}
levl_chunk;
```

Remarks

- The header only uses DWORDs (4 byte values) or multiples of 4 character bytes to avoid problems with alignment of structures in different compilers.
- The total size of the header is 128 bytes in order to avoid cache misalignment.
- There is space for extending the format.

2.2. Elements of the “levl” chunk

ckID This is the 4 character array {“l”, “e”, “v”, “l”} [1], the chunk identification.

ckSize The size of the remainder of the chunk. (It does not include the 8 bytes used by ckID and ckSize.)

dwVersion The version of the peak_envelope chunk. It starts with 0000.

1. The definition DWORD ckID = “levl” would not be unique. Different C-compilers produce different orders of the characters. Therefore we define char ckID[4] = {“l”,”e”,”v”,”l”} instead.

dwFormat The format of the peak envelope data. Two formats are allowed [2]:

dwFormat	Value	Description
LEVL_FORMAT_UINT8	1	unsigned char for each peak point
LEVL_FORMAT_UINT16	2	unsigned short integer for each peak point

dwPointsPerValue This denotes the number of peak points per peak value. This may be either one or two.

dwPointsPerValue = 1:

Each peak value consists of one peak point. The peak point is the maximum of the absolute values of the **dwBlockSize** audio samples in each block:

max{abs(X1),...,abs(Xn)}

Note: In this case the displayed waveform will always be symmetrical with respect to the horizontal axis.

dwPointsPerValue = 2:

Each peak value consists of two peak points. The first peak point corresponds to the highest *positive* value of the **dwBlockSize** audio samples in the block. The second peak point corresponds to the *negative* peak of the **dwBlockSize** audio samples in the block.

It is recommended to use two peak points (**dwPointsPerValue = 2)** because unsymmetrical wave forms (e.g. a DC offset) will be correctly displayed.

dwBlockSize This is the number of audio samples used to generate each peak frame. This number is variable. The default and recommended block size is 256.

dwPeakChannels The number of peak channels [3].

dwNumPeakFrames The number of peak frames. The number of peak frames is the integer obtained by rounding down the following calculation:

$$\text{dwNumPeakFrames} = \frac{(\text{numAudioFrame} + \text{dwBlockSize})}{\text{dwBlockSize}}$$

or rounding up the following calculation:

$$\text{dwNumPeakFrames} = \frac{\text{numAudioFrame}}{\text{dwBlockSize}}$$

Where numAudioFrame is the number of audio samples in each channel of the audio data.

E.g. for a peak ratio (Block size) of 256, this means:

0	audio sample	-> 0 peak frame
1	audio sample	-> 1 peak frame
256	audio samples	-> 1 peak frame
257	audio samples	-> 2 peak frames
7582	audio samples	-> 30 peak frames

2. Because any audio application that supports the **"levl"** chunk would have to implement all possible formats, only two formats are allowed.
In most cases the unsigned char (8 bit) format is sufficient. The unsigned short format (16 bit) should cover any cases needing higher precision.

3. Usually the number of peak channels equals the number of audio channels. If this number is one, the same waveform will be displayed for each audio channel.

dwPosPeakOfPeaks An audio application can use this information to normalize a file without having to scan the entire file. (Since has already been done by the sender). The benefit is a performance boost as well as the possibility to normalize a file in real-time.

The **peak-of-peaks** is *first* audio sample whose absolute value is the maximum value of the entire audio file.

Rather than storing the peak-of-peaks as a sample value, the *position* of the peak of the peaks is stored. In other words, an audio sample frame index is stored. An application then knows *where* to read the peak of the peaks in the audio file. It would be more difficult to store a value for peak since this is dependant on the binary format of the audio samples (integers, floats, double...).

If the value is `0xFFFFFFFF`, then that means that the peak of the peaks is unknown.

dwOffsetToPeaks Offset of the peak data from the start of the header. Usually this equals to the size of the header, but it could be higher.

This can be used to ensure that the peak data begins on a `DWORD` boundary.

strTimeStamp A C string containing the time stamp of the creation of the peak data. It is formatted as follows: [4]

"YYYY:MM:DD:hh:mm:ss:uuu"

where

YYYY:	year
MM:	month
DD:	day
hh:	hours
mm:	minutes
ss:	seconds
uuu:	milliseconds

Example: "2000:08:24:13:55:40:967"

2.3. Format of a peak point

A peak value is composed of one or two peak points, flagged by **dwPointsPerValue**. The flag **dwFormat** indicates the format of the numbers representing the peak points in each peak frame.

		dwPointsPerValue	
		= 1	= 2
dwFormat		The number corresponds to the absolute peak	The first number corresponds to the positive peak. The second number corresponds to the negative peak (Note that the "negative" peak is stored as a "positive" number.)
= 1	levl_format_uint8	*unsigned char* (0...255)	*unsigned char* (0...255) *unsigned char* (0...255)
= 2	levl_format_uint16	*unsigned short* (0...65535)	*unsigned short* (0...65535) *unsigned short* (0...65535)

4. This format has the advantage that there is no limitation in time and it is easy to read. (Other formats use a DWORD denoting the seconds since 1970, which reaches its limit after about 125 years.)

2.4. Multi channel peak files

For multichannel audio files, the single peak values from each channel are interleaved. A set of interleaved peak values is called a peak frame. The order of the peak values inside a peak frame corresponds to the placement of the sample points inside the RIFF audio data frame.

2.5. Synchronization with the audio file

The peak file must be rebuilt if either of these two conditions is met:

The time stamp is older than the time stamp of the audio file.

The number of peak frames does not correspond to the number of sample frames in the audio file

2.6. Byte order

Because the Broadcast Wave Format, BWF is an extension to the RIFF format, all numbers are stored as little-endian (Intel format).

Bibliography

[1] EBU document Tech 3285: **Specification of the Broadcast Wave Format – A format for audio files in broadcasting.**