Embedding Metadata in Digital Audio Files

Introductory Discussion for the Federal Agencies Guideline

By the Federal Agencies Audio-Visual Working Group
UUhttp://www.digitizationguidelines.gov/audio-visual/

Version 2, Approved by Working Group on March 23, 2012.

TABLE OF CONTENTS

INTRODUCTION

What is this document?

This is one of four documents pertaining to the embedding of metadata in digital audio files prepared by the Federal Agencies Audio-Visual Working Group in 2011. The three companion documents are:

- *Guideline for Federal Agency Use of Broadcast WAVE Files (Version 2.0)*
http://www.digitizationguidelines.gov/audio-visual/documents/Embed_Guideline_20120323.pdf
- *Consultant's report on embedding options in digital audio files*
http://www.digitizationguidelines.gov/audio-visual/documents/AVPS_Audio_Metadata_Overview_090612.pdf
- *Discussion paper: Identifiers: Types and Characteristics*
http://www.digitizationguidelines.gov/audio-visual/documents/IdentifiersTypesCharacteristics_20111121.pdf

Meanwhile, access is still provided to the two predecessor documents:

- Version 1.0 of the *Guideline* (September 15, 2009):
http://www.digitizationguidelines.gov/audio-visual/documents/Embed_Guideline_090915.pdf
- The September 15, 2009 version of this introduction:
http://www.digitizationguidelines.gov/audio-visual/documents/Embed_Intro_090915.pdf

Why embed metadata?

Embedded metadata can provide information to and support functionality for various persons and systems at a variety of points in the content life cycle. For example, it can help the digitizing unit or organization as it produces and preserves content. It can serve persons or systems who receive content that is disseminated by the digitizing unit or organization. Some metadata elements are especially valuable to internal actors, some to external, and some to both.

Embedded metadata, of course, is rarely an agency's only metadata. In most archiving and preservation programs, workflow and archiving are supported by one or more databases, cataloging systems, finding aids, and the like, each of which contains metadata. Many if not all metadata elements turn up in more than one place, a good thing since redundancy supports long-term preservation. (Being in more than one place, however, can make it difficult to update metadata across the board, unless this is supported in an automated way by an organization's technical infrastructure.)

Limits to the Audio-Visual Working Group's current proposal

As the larger federal agencies activity proceeds, the current investigation by the Audio-Visual Working Group is addressing only a portion—albeit a pressing and important portion—of the larger topic. Here are some of our limits:

- Format type. The associated guideline pertains to embedded metadata in audio files that result from the reformatting of analog content.
- Content lifecycle. The guideline is concerned with the initial stages of production and archiving. Of course, some of the elements identified—e.g., the embedding of appropriate identifiers—will "pay off" in later stages of the life cycle.
- Files not packages. The Working Group uses the term digital file to name a single computer file, e.g., a WAVE file. The term digital package is used to name a digital entity that represents a single intellectual or logical entity, e.g., a digital copy of a published long-playing phonograph

[1] The emphasis of the associated guideline is on the metadata to be embedded in files.

- Master files more than derivative files. The guideline is concerned with embedded data in master and production master files and, to a lesser degree, with derivative files.
- Administrative and descriptive metadata, including identifiers. The focus for the guideline is on administrative and descriptive metadata. This is not to discount the importance of technical metadata but, for audio files, this category seems to be well established. See the section "Other technical metadata" (page 12 below) for more information on the format chunk in WAVE files.

Constraints imposed by WAVE and BWF file specifications, future explorations

The options for embedding administrative and descriptive metadata are limited by the WAVE and BWF formats' specifications. The underlying structure has been inherited from the 1991 Microsoft-IBM RIFF specification[2] that calls for an extensible structure made up of chunks. A number of chunks are specified in this document, including what is called the INFO list chunk (and "subchunks") for descriptive and administrative metadata. Over the years, various organizations have added new chunks. Important chunks were added by the European Broadcasting Union (EBU) as part of its standardization of the Broadcast WAVE format (a WAVE subtype often referred to as BWF). The most widely used BWF chunk is called bext (Broadcast Extension), which allows for additional metadata to serve the needs of broadcasters. The data elements available in INFO and the bext Chunk are listed in later sections of this document.

Three other metadata chunks of interest have been established for Broadcast WAVE files. The first of these was developed by the EBU in 2003. It is an additional chunk for XML data called the axml Chunk, and a corollary schema called aXML.[3] Although of interest and potentially very useful for archiving, the axml Chunk does not seem to have been widely adopted by broadcasters and digital audio workstation manufacturers, although some audio applications like *Basehead* can read the aXML data.[4]

---

[1] This limit means that this set of guidelines will not address the specialized metadata that may be used to package multichannel (http://www.digitizationguidelines.gov/term.php?term=multitrackaudio), multitrack (http://www.digitizationguidelines.gov/term.php?term=multitrackaudio), and multisegment recordings. *Multichannel* and *multitrack* are defined by the glossary entries in the URLs cited; *multisegment* refers to such things as single, lengthy performances that have been broken into segments, each of which is represented by its own file. This topic receives insightful discussion in the final report from the Sound Directions project carried out by Indiana and Harvard Universities; see http://www.dlib.indiana.edu/projects/sounddirections/papersPresent/index.shtml.

[2] RIFF is documented in *Multimedia Programming Interface and Data Specifications 1.0* (1991, IBM Corporation and the Microsoft Corporation). This document is available at several Web sites, including http://www.kk.iij4u.or.jp/~kondo/wave/mpidata.txt and http://www.tactilemedia.com/info/MCI_Control_Info.html.

[3] "Specification of the Broadcast Wave Format; A Format for Audio Data Files in Broadcasting; Supplement 5: <axml> Chunk," http://tech.ebu.ch/docs/tech/tech3285s5.pdf

[4] From the Basehead Web site (http://www.baseheadinc.com, consulted November 10, 2011): "BaseHead is our monster product for searching and finding your Sound Effects, Music and Audio Files. It already dominates the Video Game Industry . . . ."

The second metadata chunk, also for XML data, was developed by a British trade group, the Institute of Broadcast Sound (IBS).[5] The current version of the specification is dated 2010; earlier versions of the specification can be found in the Internet Archive Wayback machine from 2004 forward.[6] The accompanying report[7] from AudioVisual Preservation Solutions describes the iXML chunk as having been "created by group of audio hardware and software manufacturers in order to facilitate transfer of production metadata across systems and is offered as an expansion of the bext chunk."[8] iXML is not just a chunk, it is also an extensible XML schema designed for use in audio production and editing.[9] As of this writing, the iXML website lists 16 companies and 23 products that use the data structure, and iXML is also being developed as a standard by the Audio Engineering Society (AES). Standardization by the AES could lead to greater adoption outside the audio production community.

The third metadata chunk is an implementation of Adobe's XMP specification. This structure can exist as a "sidecar" file or be embedded in a variety of content-essence files, including WAVE (need not be Broadcast WAVE).[10] XMP has been widely adopted by professional photographers because it is well supported in the ubiquitous family of Adobe imaging software, e.g., Photoshop. XMP is frequently embedded in TIFF, JPEG, DNG, PDF, and other image formats. For photographs, the XMP data is generally structured to comply with a news-oriented metadata standard from International Press and Telecommunications Council (IPTC).[11] Although Adobe and the IPTC are actively exploring ways to expand the use of XMP among audio and video producers, especially in the field of journalism, the broad array of companies who market audiovisual production and editing software may or may not embrace what is viewed as an Adobe specification.

On paper, these chunks overcome many of the limitations of the BWF bext Chunk and the RIFF INFO List. Their low levels of adoption in the archival world at this time, however, make them less appealing than one would wish. Nevertheless, the Working Group will explore them in the future as possible improved solutions to the metadata-embedding needs of preservation-oriented archives.

---

[5] From the IBS Web site (http://www.ibs.org.uk/, consulted November 10, 2011): "The IBS was founded in 1977 by sound balancers in radio and television, who felt a need for a better interchange of ideas between practitioners in the various areas of broadcast audio."

[6] The URL for the 2010 specification is http://www.ixml.info/ (consulted November 10, 2011). The earliest Wayback instance is version 1.27 dated October 9, 2004; http://web.archive.org/web/20040924101734/http://www.ixml.info/ (consulted November 10, 2011).

[7] http://www.digitizationguidelines.gov/audio-visual/documents/AVPS_Audio_Metadata_Overview_090612.pdf

[8] IBID

[9] For more information about iXML see http://www.gallery.co.uk/ixml/

[10] The URL for the embedding part of the XMP specification is http://wwwimages.adobe.com/www.adobe.com/content/dam/Adobe/en/devnet/xmp/pdfs/XMPSpecificationPart3.pdf (consulted November 10, 2011).

[11] See http://www.iptc.org/IPTC4XMP/ (consulted November 10, 2011).

DISCUSSION

Versioning and Broadcast WAVE

There have been three iterations of the BWF format under the general specification number EBU Tech 3285. Version 0 (as it came to be called) was published in 1997. Version 1, which differs from Version 0 in the use of 64 of the 254 reserved bytes to contain a SMPTE UMID identifier, was published July 2001. Version 1 was followed by six supplements with additional information. Version 2 references these supplements and adds elements that pertain to loudness metadata.

Because Version 0 files do not have a space reserved for the UMID and neither Version 0 or Version 1 files have spaces for loudness metadata, the writers of this document have concocted a simple rubric for determining the version of a Broadcast WAVE file:

- If a file has BEXT data but no UMID or loudness values, it is Version 0. (The value for the BEXT Version element is not specified by EBU; FADGI recommends using 0000h)
- If a file has BEXT data and a UMID value but no loudness values, it is Version 1. (EBU specifies 0001h as the value for the Version element.)
- If a file has BEXT data and loudness values, it is Version 2. (EBU specifies 0002h as the value for the Version element.)

What are the potential problems in entering the wrong version for a file? We do not foresee problems beyond the possibility that a Version 2-compliant piece of software may not see and/or act upon the loudness metadata in a file that does not declare itself as Broadcast WAVE Version 2.

As for compatibility with respect to Version 2, the specification itself has this to say:

"Version 2 is a substantial revision of Version 1 which incorporates loudness metadata (in accordance with EBU R 128 [2]) and which takes account of the publication of Supplements 1 – 6 and other relevant documentation. This version is fully compatible with Versions 0 and 1, but users who wish to ensure that their files meet the requirements of EBU Recommendation R 128 will need to ensure that their systems can read and write the loudness metadata."

Data Elements in the WAVE (RIFF) INFO List

The INFO chunks are inherited by WAVE and BWF from the specification for their parent, the Resource Interchange File Format (RIFF).[12] The following descriptive overview was collected on December 22, 2008, from an online document titled "Inside the RIFF Specification" by Hamish Hubbard, dated September 1, 1994.[13].

- LIST chunks are the only chunks apart from RIFF chunks that may contain their own subchunks . . . . LIST chunks are usually subchunks of RIFF chunks themselves. Like RIFF

[12] See http://www.digitalpreservation.gov/formats/fdd/fdd000025.shtml
[13] http://drdobbs.com/database/184409308

chunks, LIST chunks have a four-character code in the first four bytes of their data area. This code specifies the list type (analogous to a RIFF chunk's form type) . . . .

- For example, a LIST chunk of list type INFO may contain subchunks such as INAM (the name of the data stored in the file) and ICRD (creation date). LIST chunks of type INFO are optional in current RIFF forms, but their use is recommended. The LIST chunks' subchunks can store much more information about the file than is available from the filename and date stamp. These LIST subchunks share a common format: Each contains one ASCII Z (NULL terminated) string.

The following introductory statement and element list is from the RIFF specification, the Multimedia Programming Interface and Data Specifications 1.0, issued as a joint design by IBM Corporation and Microsoft Corporation, August 1991. In this document the block of data is called *INFO list chunk* (singular) and in another section there is a reference to the *list chunk type* as having *subchunks*. Regarding the "values" that can be associated with each subchunk or element, the writers of this document believe that there is no particular limit on the length of the null-terminated coded chunks. We also believe that these subchunks are not repeatable. Comments from specialists in the field will be welcome.

The INFO list is a registered global form type that can store information that helps identify the contents of the chunk. This information is useful but does not affect the way a program interprets the file; examples are copyright information and comments. An INFO list is a LIST chunk with lis type INFO. The following shows a sample INFO list chunk:

```
LIST('INFO'   INAM("Two Trees"Z)
          ICMT("A picture for the opening screen"Z) )
```

An INFO list should contain only the following chunks. New chunks may be defined, but an application should ignore any chunk it doesn't understand. The chunks listed below may only appear in an INFO list. Each chunk contains a ZSTR, or null-terminated text string.

Chunk ID	**Description**
IARL	Archival Location. Indicates where the subject of the file is archived.
IART	Artist. Lists the artist of the original subject of the file. For example, Michelangelo.
ICMS	Commissioned. Lists the name of the person or organization that commissioned the subject of the file. For example, Pope Julian II.
ICMT	Comments. Provides general comments about the file or the subject of the file. If the comment is several sentences long, end each sentence with a period. Do not include newline characters.
ICOP	Copyright. Records the copyright information for the file. For example, Copyright Encyclopedia International 1991. If there are multiple copyrights, separate them by a semicolon followed by a space.
ICRD	Creation date. Specifies the date the subject of the file was created. List dates in year-month-day format, padding one-digit months and days with a zero on the left. For example, 1553-05-03 for May 3, 1553.
ICRP	Cropped. Describes whether an image has been cropped and, if so, how it was cropped. For example, lower right corner.

IDIM	Dimensions. Specifies the size of the original subject of the file. For example, 8.5 in h, 11 in w.
IDPI	Dots Per Inch. Stores dots per inch setting of the digitizer used to produce the file, such as 300.
IENG	Engineer. Stores the name of the engineer who worked on the file. If there are multiple engineers, separate the names by a semicolon and a blank. For example, Smith, John; Adams, Joe.
IGNR	Genre. Describes the original work, such as, landscape, portrait, still life, etc.
IKEY	Keywords. Provides a list of keywords that refer to the file or subject of the file. Separate multiple keywords with a semicolon and a blank. For example, Seattle; aerial view; scenery.
ILGT	Lightness. Describes the changes in lightness settings on the digitizer required to produce the file. Note that the format of this information depends on hardware used
IMED	Medium. Describes the original subject of the file, such as, computer image, drawing, lithograph, and so forth.
INAM	Name. Stores the title of the subject of the file, such as, Seattle From Above.
IPLT	Palette Setting. Specifies the number of colors requested when digitizing an image, such as 256.
IPRD	Product. Specifies the name of the title the file was originally intended for, such as Encyclopedia of Pacific Northwest Geography.
ISBJ	Subject. Describes the contents of the file, such as Aerial view of Seattle.
ISFT	Software. Identifies the name of the software package used to create the file, such as Microsoft WaveEdit.
ISHP	Sharpness. Identifies the changes in sharpness for the digitizer required to produce the file (the format depends on the hardware used).
ISRC	Source. Identifies the name of the person or organization who supplied the original subject of the file. For example, Trey Research.
ISRF	Source Form. Identifies the original form of the material that was digitized, such as slide, paper, map, and so forth. This is not necessarily the same as IMED.
ITCH	Technician. Identifies the technician who digitized the subject file. For example, Smith, John.

Data elements in the BWF bext Chunk

The list of elements and a descriptive definition that follows is taken from this standards document from the European Broadcasting Union (EBU): "BWF – A Format for Audio Data Files in Broadcasting," ver. 2.0, Tech 3285 (Geneva: Switzerland: European Broadcasting Union, May, 2011).[14] The limits on the extent of the values permitted for each tag are provided in the definitions. The writers' understanding is that these fields are not repeatable.

Description.	ASCII string (maximum 256 characters) containing a free description of the sequence. To help applications which only display a short description, it is recommended that a résumé of the description is contained in the first 64 characters, and the last 192 characters are use for details. If the length of the string is less than 256 characters, the last one is followed by a null character

[14] http://tech.ebu.ch/docs/tech/tech3285.pdf

	(00)
Originator.	ASCII string (maximum 32 characters) containing the name of the originator/producer of the audio file. If the length of the string is less than 32 characters, the field is ended by a null character.
OriginatorReference.	ASCII string (maximum 32 characters) containing a non ambiguous reference allocated by the originating organization. If the length of the string is less than 32 characters, the field is ended by a null character. Note: The EBU has defined a format for the OriginatorReference field. See EBU Recommendation R99-1999.
OriginationDate.	Ten ASCII characters containing the date of creation of the audio sequence. The format is .yyyy-mm-dd. (year-month-day). Year is defined from 0000 to 9999; Month is defined from 1 to 12; Day is defined from 1 to 28,29,30 or 31. The separator between the items can be anything but it is recommended that one of the following characters is used: .-. hyphen ._. underscore .:. colon . . space ... stop [Editor's note: it is not possible to use the syntax of ISO 8601 datetime for this element. Thus date and time cannot be expressed in terms of UTC.]
OriginationTime.	Eight ASCII characters containing the time of creation of the audio sequence. The format is .hh-mm-ss. (hours-minutes-seconds). Hour is defined from 0 to 23. Minute and second are defined from 0 to 59. The separator between the items can be anything but it is recommended that one of the following characters is used: .-. hyphen ._. underscore .:. colon . . space ... stop [Editor's note: it is not possible to use the syntax of ISO 8601 datetime for this element. Thus date and time cannot be expressed in terms of UTC.]
TimeReference.	This field contains the timecode of the sequence. It is a 64-bit value which contains the first sample count since midnight. The number of samples per second depends on the sample frequency which is defined in the field <nSamplesPerSec> from the <format chunk>. [Editor's note: This metadata element records what is sometimes called a *timestamp*, i.e., the start time for a given file, in terms of a timeline of the sort often provided by digital audio workstation software. Timelines are often set up to begin at 00:00:00

[15] The acronym LUFS stands for *Loudness Unit referenced to Full Scale*. We have taken this term from EBU – Recommendation R 128 (http://tech.ebu.ch/docs/r/r128.pdf), which notes that LUFS is compliant with international naming conventions and is equivalent to LKFS, as used in the important ITU-R BS.1770-2 specification pertaining to measurement (http://www.itu.int/dms_pubrec/itu-r/rec/bs/R-REC-BS.1770-2-201103-I!!PDF-E.pdf). These terms have been defined with slight variations by different standards group; see the 2009 *Note on measurement units for loudness* by Søren H. Nielsen (http://www.tcelectronic.com/media/nielsen_loudness_units.pdf, consulted November 10, 2011). There is also a 2010 working document (not publicly available) titled 'Proposal for the rationalisation of nomenclature used in ITU R BS.1770 and ITU-R BS.1771' (see http://www.itu.int/md/R07-WP6C-C-0324/en, consulted November 10, 2011).

	("midnight"). Files can be placed on the timeline in terms of their sequence using TimeReference. In <bext>, the time value is provided in terms of sample count for timeline location for the first sample in the file. The Working Group understands that this element has special value in the case of multitrack or multisegment content, to be explored further at another time.]
Version.	An unsigned binary number giving the version of the BWF. This number is particularly relevant for the carriage of the UMID and loudness information. For Version 1 it shall be set to 0001h and for Version 2 it shall be set to 0002h.
UMID.	64 bytes containing a UMID (Unique Material Identifier) to standard SMPTE 330M. If only a 32 byte "basic UMID" is used, the last 32 bytes should be set to zero. (The length of the UMID is given internally.)
Reserved.	190 bytes reserved for extensions. If the Version field is set to 0001h, these 190 bytes must be set to a NULL (zero) value.
CodingHistory.	Non-restricted ASCII characters, containing a collection of strings terminated by CR/LF. Each string contains a description of a coding process applied to the audio data. Each new coding application is required to add a new string with the appropriate information. This information must contain the type of sound (PCM or MPEG) with its specific parameters: -- PCM : mode (mono, stereo), size of the sample (8, 16 bits) and sample frequency: -- MPEG : sample frequency, bit-rate, layer (I or II) and the mode (mono, stereo, joint stereo or dual channel). It is recommended that the manufacturers of the coders provide an ASCII string for use in the coding history. Note: The EBU has defined a format for CodingHistory which will simplify the interpretation of the information provided in this field. See EBU Recommendation R98-1999.
LoudnessValue	A 16-bit signed integer, equal to *round* (100x the Integrated Loudness Value of the file in LUFS).[15]
LoudnessRange	A 16-bit signed integer, equal to *round* (100x the Loudness Range of the file in LU).
MaxTruePeakLevel	A 16-bit signed integer, equal to *round* (100x the Maximum True Peak Value of the file in dBTP)>
MaxMomentaryLoudness	A 16-bit signed integer, equal to *round* (100x the highest value of the Momentary Loudness Level of the file in LUFS).

MaxShortTermLoudness A 16-bit signed integer, equal to *round* (100x the highest value of the Short-term Loudness Level of the file in LUFS).

Note: Loudness and related metadata is discussed below.

The *CodingHistory* element

The Coding History element is interesting because it offers a place and a method for recording process or digital-provenance metadata. The use of this field to record some of the reformatting history for given item has been outlined in the final report from the Sound Directions project carried out by Indiana and Harvard Universities.[16] The report describes the use of Coding History at Indiana University and their discussion is presented in the box that follows.

The Coding History field is designed to hold data on the digitizing process including signal chain specifics, sample rate and bit depth, and other elements. It is defined as a collection of strings, each presented on a separate line, containing a history of the coding processes applied to the file. Each variable within a string is separated by a comma. A new line is added when the coding history related to the file is changed, and each line should end with a carriage return and line feed which are automatically added by WaveLab. According to the EBU, each line should contain these elements, as appropriate to the coding history being described:
-- Coding algorithm. String begins with "A=" For example: A=ANALOG, PCM, MPEG1L3, and others
-- Sampling frequency. String begins with "F="
-- Bit-rate, for MPEG coding only. String begins with "B="
-- Word length. String begins with "W="
-- Mode—this corresponds to sound field, such as mono, stereo, or dual-mono. String begins with "M="
-- Text, free string—a free ASCII-text string for in-house use. The EBU suggests documenting devices in the signal chain and analog source recording formats in this field. String begins with "T="

At Indiana, we include three lines of coding history in our BWF files for the digitization of analog recordings. The first documents the analog source recording, the second contains data on digitization chain, while the third records information on the storage of the file.

For example:
A=ANALOG,M=mono,T=Studer A810; SN3690; 15 ips; open reel tape,
A=PCM,F=96000,W=24,M=mono,T=Benchmark; ADC1; SN00252; A/D,
A=PCM,F=96000,W=24,M=mono,T=Lynx; AES16; DIO,

Line 1 reads: an analog open reel tape with a mono sound field was played back on a Studer A810 tape machine with serial number 3690. Tape speed was 15 ips.

[16] http://www.dlib.indiana.edu/projects/sounddirections/papersPresent/index.shtml

While the EBU document suggests including the tape brand and product number as the last element, we prefer a general designation of the format for several reasons: it is more useful to know the format than the specific brand and it avoids the need to interpret the brand information and playback machine data to identify the format. When a range of formats—analog cassettes, discs, DATs and others—are routinely digitized this interpreting might become unnecessarily difficult. In addition, the format remains constant through an entire collection (the brand and product number may or may not), providing one less element that requires data entry for each source recording.

Line 2 reads: the tape was digitized in mono mode using a Benchmark ADC1 A/D converter with serial number 00252 at 96 kHz sample rate with a bit depth of 24 bits.

Line 3 reads: the tape was stored as a 96/24 mono file using a Lynx AES16 digital input/output interface.

If we apply additional coding processes to produce a derivative file we add a fourth line in the header of the derivative file. For example:

A=PCM,F=44,100,W=16,M=mono,T=Steinberg; WaveLab 6; Resampler, Waves L2; Dither; DAW,

This line reads: A 16 bit, 44.1 kHz file was created using the WaveLab 6 Resampler and Waves L2 Dither in the Digital Audio Workstation.

The *loudness* elements and the potential use of loudness metadata by heritage institutions

The most significant new development in the BWF standard and, by extension, in the Federal Agencies Working Group's revised guideline, is the inclusion of metadata elements pertaining to the measurement of *loudness* in the file. The European Broadcast Union (EBU) added these fields to version 2 of their Broadcast Wave Format specification in May, 2011. The EBU document *Loudness normalisation and permitted maximum level of audio signals*[17] lists the reasons for the change, including the following:

- peak normalization of audio signals has led to considerable loudness differences between programs;
- the resulting loudness inconsistencies are the cause of the most viewer/listener complaints;
- an international standard for measuring audio programme loudness has been defined in ITU-R BS.1770, introducing the measures LU (Loudness Unit) and LUFS (Loudness Unit, referenced to Full Scale);[18]
- a gated measurement of *Programme Loudness* (hence measuring 'Foreground Loudness') is advantageous to improve the loudness matching of programs with a wide loudness range;

In addition to broadcasting, there is interest in loudness in the recording industry. Industry commentators have written about "loudness wars," i.e., the continuing one-upmanship between

[17] EBU R 128, August 2011, http://tech.ebu.ch/docs/r/r128.pdf

[18] See footnote 16 above, concerning the definitions of LUFS and LKFS.

recording labels to produce successively louder analog and, later, digital music releases. With the advent of the iPod these loudness wars have gotten even more aggressive with record labels. EBU-loudness-compliant audio files could, if played back in loudness-compliant software, be normalized so that no programs or musical selections are objectionably louder than the others. This could also be applied to broadcast settings, obviating the potential for listener discomfort at strikingly louder commercials or other content.

The EBU and ITU-R's application of loudness metrics represents a shift in how audio levels are measured and declared. The traditional measure has focused on *peaks*, the moments of high volume that must be managed to prevent over-modulation or clipping. For many years, the VU meter was used to measure audio peaks. VU meters are a "slow" measurement that averages the peaks and troughs of a sound recording. One of the challenges of the peak meter, according to the Wikipedia article *Peak meter*,[19] is that "*because of the mass of the moving parts and mechanics, the response time of these older meters could have been anywhere from a few milliseconds to a second or more. Thus, the meter might not ever accurately reflect the signal at every instant of time, but the constantly changing level, combined with the slower response time, led to more of an 'average' indication."* In other words, if there are signal peaks or troughs of shorter duration than the peak meters' response time, they may not be accurately represented. Over time, various refinements were sought. The Wikipedia article *Peak program meter*[20] describes some developments: the true peak programme meter, the quasi-peak programme meter (QPPM), the sample peak programme meter (SPPM), and the over-sampling peak programme meter. It is worth noting that replacing VU and peak meters with loudness measurements does not imply that these measuring devices were not high-quality, standardized pieces of equipment. Their limitations stemmed from the limitations of human perception – because the needle-based displays were intended to be eye-legible, they could only move so fast before they became a blur. There are also limits on how quickly the human ear can integrate sounds. While VU and peak meters could measure sounds with considerable precision, the EBU's loudness measurements could potentially be more precise by obviating (or circumventing) the recordist's perceptual limitations.[21]

Meanwhile, professionals noted that peak readings were not sufficient to manage sound volume in terms of human perception. Loudness metrics are a response to this technical difficulty. The measurements are designed to be a "practical solution for the task of finding an objective measurement of what is essentially a subjective impression (loudness)."[22] The document *EBU R 128: Audio loudness normalization and permitted maximum level of audio signal* uses the terms *Programme Loudness*, *Loudness Range* and *Maximum True Peak Level* to describe audio signals. The definitions of these terms, from *EBU R 128*, are below:

---

[19] http://en.wikipedia.org/wiki/Peak_meter, consulted on September 13, 2011.
[20] http://en.wikipedia.org/wiki/Peak_programme_meter, consulted on September 13, 2011.
[21] Richard Wright of the BBC reminds us that "the ITU 'loudness meter' also integrates power over a defined time,... The 'new' method integrate[s] digitally with a rectangular window instead of exponential decay as in an analogue meter (achieved with hardware by use of a resistor and a capacitor; it wasn't the meter movement itself that defined the response time) and the 'new' approach uses memory and processing to accumulate a sequence of 'short-term integrations of power' over an entire audio file, and then do statistical processing to 'gate' the levels that are too low to represent 'signal of interest.'" Email correspondence, November 1, 2011.
[22] Camerer, Florian. "Overview of the EBU Loudness Recommendation R 128." *SMPTE Motion Imaging Journal*. July-August 2011; 120:(5) 24-28.

- **Programme Loudness:** The integrated loudness over the duration of a programme - Programme Loudness Level is the value (in LUFS [Loudness Unit referenced to Full Scale]) of Programme Loudness

- **Loudness Range (LRA):** This describes the distribution of loudness within a programme;

- **Maximum True Peak Level:** The maximum value of the audio signal waveform of a programme in the continuous time domain.

As indicated above, in contrast to the various means of measuring the peaks of an audio program, these new values look at more dimensions related to the loudness of the entire program. How might loudness metrics pertain to such activities as the preservation reformatting of older sound recordings? The current practice of audio transfers in a preservation setting is to do "flat" transfers – that is, to set a level so the highest peaks of an audio program do not clip and keeping to that level throughout the transfer. This differs from the broadcasters' practice of "riding the levels," meaning they continually adjust the levels of a program as it is being broadcast in order to avoid clipping the peaks and to bring up quieter passages.

The authors of this document believe that the capture of loudness metrics will not affect the preservation transfers of historical recordings. That is, we expect that the practice of setting the level safely below the highest peak--and not "riding" this level through the transfer--will continue. However, as new tools become available, loudness metadata may be embedded in the file, thus permitting a "smart" playback device to adjust the playback volume to suit the preferences of the listener.

Some tools that can provide loudness metrics are coming into the marketplace and we cannot see any reason for archives not to use them to create loudness metadata and add it to the bext Chunk. However, the newness of loudness metrics and our community's lack of experience with them leave us uncertain as to how archives, museums, and libraries will move these ideas into practice. The authors of this guidelines document asked some colleagues in the cultural heritage community if they thought these loudness elements would or could be important in a library/archives/museum preservation setting. The replies indicated that there could be value in recording this kind of information for reasons similar to that cited by broadcasters--to make the listening experience easier on the user--but for now, it is probably not critical information in a preservation setting. For this reason we have included the loudness elements as "optional" in our guideline for the bext Chunk.

It is worth noting that archives do not only produce recordings, e.g., preservation copies of historical materials. At least some archives in the federal sector, and many in other sectors, will begin to acquire loudness-compliant digital audio materials in the foreseeable future. If these born-digital acquisitions consist of Broadcast WAVE files with embedded loudness metadata, the archives will surely wish to retain that metadata. This means that archives' data management toolsets must be loudness-aware in order to properly read and manage such files. This is one of the reasons that we are updating the open source tool BWF MetaEdit (http://sourceforge.net/projects/bwfmetaedit/) to accommodate loudness metadata.

Other technical metadata in WAVE files: the WAVE Format Chunk

The key document that governs WAVE-specific technical metadata is the RIFF (Resource Interchange File Format) specification: *Multimedia Programming Interface and Data Specifications 1.0* (1991, IBM Corporation and the Microsoft Corporation). This publication specifies the content for the WAVE Format Chunk (<fmt-ck>), which in turn specifies documents the format of the actual waveform sound data (<wave-data>).

Field	Description
FormatTag	A number indicating the WAVE format category of the file. The content of the <format-specific-fields> portion of the fmt chunk, and the interpretation o the waveform data, depend on this value. You must register any new WAVE format categories. See Registering Multimedia Formats in Chapter 1, Overview of Multimedia Specifications, for information on registering WAVE format categories. Wave Format Categories, following this section, lists the currently defined WAVE format categories. If the wFormatTag field of the <fmt-ck> is set to WAVE_FORMAT_PCM, then the waveform data consists of samples represented in pulse code modulation (PCM) format.[23] For PCM waveform data, the <format-specific-fields> includes the BitsPerSample data element (next item listed).
BitsPerSample	Specifies the number of bits of data used to represent each sample of each channel. If there are multiple channels, the sample size is the same for each channel.
Channels	The number of channels represented in the waveform data, such as 1 for mono or 2 for stereo.
SamplesPerSec	The sampling rate (in samples per second) at which each channel should be played.
AvgBytesPerSec	The average number of bytes per second at which the waveform data should be transferred. Playback software can estimate the buffer size using this value
BlockAlign	The block alignment (in bytes) of the waveform data. Playback software need

[23] The *Multimedia Programming Interface and Data Specifications 1.0* lists the following WAVE format categories with their FormatTag, Value, and Format Category:
WAVE_FORMAT_PCM (0x0001) Microsoft Pulse Code Modulation (PCM) format
IBM_FORMAT_MULAW (0x0101) IBM mu-law format
IBM_FORMAT_ALAW (0x0102) IBM a-law format
IBM_FORMAT_ADPCM (0x0103) IBM AVC Adaptive Differential Pulse Code Modulation format
Many additional categories are lists in the documentation for the JHOVE WAVE module (http://hul.harvard.edu/jhove/wave-hul.html).

	to process a multiple of BlockAlign bytes of data at a time, so the value of BlockAlign can be used for buffer alignment.

Audio-data checksums, an ad hoc MD5 Chunk, and the EBU Quality Chunk

When the BWF MetaEdit tool was being developed in 2009, the consulting team at Audiovisual Preservation Solutions added a checksum (hash value) function. The rationale is provided in the help text for the tool:

> A checksum is essentially a fingerprint for a given file used for data integrity monitoring. An MD5 is one type of checksum. When an MD5 checksum is generated it produces a 32 character value which represents a unique code (or fingerprint) called a hash value, which is specific to that file. If any changes are made to that file and a checksum is generated again it will produce a different 32 character value. If nothing in a file changes and a checksum is generated again, the values will be the same.

The checksum under discussion here pertains to the audio data within the file. This limit means that an archive can update a file's descriptive metadata without disturbing the audio data proper and thereby retain the validity of the checksum. This approach is outlined in the help text:

> A traditional whole-file checksum would be altered every time BWF MetaEdit adds or edits metadata in the file. Therefore a whole-file checksum does not help with verifying the integrity of the audio within the file. While the metadata is expected to change, the audio data is not. For this reason BWF MetaEdit supports the generation of an audio-data-only checksum (including the entire <data> chunk, excluding the chunk id, size declaration, and any optional padding byte). This will create a hash value for only the audio portion of the file which helps validate the integrity of the audio but allows for alteration of the metadata.

The help menu of BWF MetaEdit also describes how and where it places the checksum within the file:

> BWF MetaEdit includes an Option called 'Evaluate MD5 for audio data'. When this is enabled BWF MetaEdit will generate an MD5 checksum for the audio data of any file that is opened and populate the *MD5Evaluated* column of the Technical View [a report associated with the tool].
>
> Another option called 'Embed MD5 for audio data' will generate an MD5 checksum for the audio data and then store it directly within the file in an MD5 chunk within the file using the id <MD5 >. The declared size of this chunk is always 16 bytes.
>
> When BWF MetaEdit opens an audio file it will immediately display any checksum stored in the <MD5 > chunk in the *MD5Stored* column. If open audio files already include an <MD5 > chunk then running 'Evaluate MD5 for audio data' will re-evaluate the checksum and display the results in the *MD5Evaluated* column. This will demonstrate any conflicts between the stored checksum and the newly evaluated checksum in order to verify the integrity of the audio data. A discrepancy between the stored and evaluated checksum indicates that the audio data was somehow altered since the last checksum was embedded either through editing or error. If you wish to overwrite an existing embedded checksum

value with a newly generated checksum value make sure that 'Embed MD5 for audio data' is selected."

The MD5 Chunk is an invention of the Working Group and the tool's developers. We feel that it has value but, by definition, it is an ad hoc, non-standardized entity. We welcome comments on how this idea might be modified or brought forward into wider practice.

In the course of the development of the MD5 chunk, we considered using the EBU Quality Chunk, defined in Supplement 2 to the Broadcast Wave Specification (July 2001).[24] The Quality Chunk has a pair of elements that document what is called the FileSecurityCode, a code that plays a role similar to the role played by our MD5 checksum.

- FileSecurityReport: This field contains the FileSecurityCode of QualityChunk. It is a 32 bits value which contains the checksum [0….231].
- FileSecurityWave: This field contains the FileSecurityCode of BWF Wave data. It is a 32 bits value which contains the checksum [0….231].

However, since the extent of the MD5 checksum is 128 bits, it will not fit in the space allotted to the FileSecurity tags. We considered other elements within the Quality Chunk with space for more characters, e.g., *Title*, but had concerns about causing interoperability and validation problems and did not adopt that approach.

[24] "BWF – A Format for Audio Data Files in Broadcasting. Supplement 2 – Capturing Report." http://tech.ebu.ch/docs/tech/tech3285s2.pdf (accessed October 13, 2011)