Federal Agencies Audio Visual Digitization Working Group

Task 5.4: Assess Options for Embedding Metadata in WAVE Files and Plan the Audio Metadata File Header Tool Development Project

Assessment Report and Initial Recommendations

AudioVisual Preservation Solutions, Inc.
350 7th Ave.
Suite 1603
New York, New York
10001
Tel: 917.548.8632 | Fax: 866.264.4275
www.avpreserve.com

Submitted June 12th, 2009

Table of Contents

Background and General Description

The project to assess options for embedding metadata in WAVE Files and to plan the Audio Metadata File Header Tool Development Project emerged from identification and documentation of needs among organizations reformatting audio collections, either internally or externally (outsourced), from legacy audio media to file-based formats. The Federal Agencies Digitization Guidelines Working Group, including the Library of Congress, National Archives and Records Administration, Smithsonian Institution, and others, seeks to identify and implement a common descriptive metadata standard and develop corresponding tools to allow the embedding of metadata into audio files. The Working Group's Web site URL is http://www.digitizationguidelines.gov/

The Working Group made a preliminary analysis that concluded that the current metadata fields and embedding tools available within the audio industry did not meet all of their needs. As an example, the bext chunk maintains 2 different identifier fields – Originator Reference and UMID. Both of these fields only allow for one identifier to be entered and one of these fields (UMID) is specific to an identifier that is not used by Federal Agencies. The participants in the Federal Agencies Digitization Working Group wish to embed multiple identifiers beyond what the bext chunk or LIST INFO chunk offer. In addition, the Working Group identified a need for one or more tools to embed the working group's recommended core metadata set (see core metadata set as described in the Working Group's proposal at temporary URL: http://home.comcast.net/~cfle/AVdocs/Embed_Guideline_090605.doc) into WAVE files.

The project to address the group's needs is seen as a multi-task activity that will begin with planning and then move to development. In addition to this assessment document, the planning phase will generate:

- a description of the salient specifications on the Working Group's core metadata needs, the requirements for the tools, and functionality to be achieved,

- a draft activity plan that suggests what might be the resource requirements and timelines for an activity to address the needs.

Immediate and Mid-Term Goals

The Working Group recognizes that there is a future activity--hoped to occur in the mid-term--that will develop an approach and tools to accommodate a block of metadata that is more extensive than that described in the Working Group's proposed guideline1. In part, this assessment document provides guidance on some options for structuring this extensive block of metadata, e.g., the information about iXML and aXML.

The immediate need, however, is to solve the problem of what and how to embed and retrieve metadata from the bext and LIST INFO chunks into WAVE files to meet the Working Group's recommendations. The activity plan being drafted will address the immediate need for a documented methodology using available tools to begin embedding their core metadata in WAVE files. The Working Group anticipates that the immediate path will include utilization of the LIST INFO and bext chunks, which, due to their limitations, involves some compromises. They also anticipate that this methodology may use the chunks' fields in ways other than originally defined or suggested by the creators of the guidelines for these chunks. The Working Group plans to disseminate information that documents and explains their decisions.

Tools for Embedding - Singular and Batch Mode and Intermediate Data Files

The Working Group's proposal is based on the assumption that an archive will have existing data, e.g., from a database, xml file, tab delimited text file, or spreadsheet, and that selected elements from that pre-existing data will be used to populate WAVE files in an automated fashion. Given the diversity of data sources in type, extent and architecture used across the federal agencies it is recommended that the Working Group standardize one (or more) files to serve as a common conformance point regardless of the data source. A conformance point file

is understood to mean a block of data that is "reported out" of the existing data source in a standardized structure. The resulting output will then serve as an intermediate between the database, where the authoritative metadata records are stored, and the tool used to embed this data into the corresponding audio file. Using a standardized data document as a conformance point file will allow the Working Group to validate the conformance point file and help guide the design of tools involved in mapping to and from this data.

The Working Group envisions two differing methods for migrating the data in the conformance point file into WAVE files, using two distinct workflows.

1. Batch embedding and retrieval of metadata from existing WAVE files. One application for a batch tool includes working with backlogs of WAVE files generated from past digitization projects. Another application would be to accommodate workflows which involve the embedding/retrieval of metadata as an automated batch post-process.
2. Singular embedding and retrieval of metadata at the point of reformatting from physical legacy audio media to WAVE files.

Tool Objectives

- Full support of the Working Group's core metadata set including fields and associated data dictionary, beginning with the bext and LIST INFO elements and, in a later phase, moving to an aXML implementation.
- Full conformance with the Working Group's core metadata functional requirements as set forth by the Working Group, beginning with the bext and LIST INFO elements and, in a later phase, moving to an aXML implementation.
- Use of existing/emerging standards for embedding of metadata in files
- Automated embedding of metadata into WAVE files from existing data sources.
- Ability to import data from various types of sources

- Simple graphical user interface (command line tool okay for intermediate step)
- Batch mode and singular mode
- Efficient processing (taking as little time as possible to alter/embed the metadata and save the file)
- Cross-platform
- Well documented (within source code and additional external documentation)
- Quality Control measures protecting/notifying of existing metadata
- Quality Control measures preventing/undoing inaccurate batch processing actions
- Creation of a log file reporting on the processes performed, files processed and issues encountered.

Evaluation Criteria of Potential Audio Metadata Standards

The development necessary in order to meet the Working Group's goals requires the comparative analysis of the specifications, limits and potential of the file formats, metadata implementations, and standards involved. This report will evaluate a list of potential metadata standards and specifications that could be implemented to store the Working Group's core metadata standard within the individual audio files. It is worth noting that the following information is intended to support both the immediate need (understood as an implementation for the bext and LIST INFO chunks) and also the mid-term need (understood to mean a future structure for embedded metadata that is better suited to the Working Group's needs). Each standard will be evaluated according to the following criteria:

- Size: limitations on the size of the chunk and character length of the values,
- Definition: flexibility and limitations on how metadata fields are defined (as an example an XML document would be considered to offer a highly flexible method of defining the data since it is self-descriptive and may be designed to use any terms or hierarchical structure necessary to define the data). Limitations of how the data may be defined could cause a loss of semantic meaning as the metadata is migrating into the chunk.
- Adoption: ubiquity and extent of software support,
- Authority: the authority of the standard or specification, extent of documentation, whether supported by corporations or standards organizations and whether that support is currently active
- Extensibility: the ability to extend the existing standard
- Storage: requirements on where the chunk may be placed within the RIFF structure and pre-requisites for the use of the chunk

Review of Fixed Standards

RIFF file structure

The RIFF specification[1], first specified by Microsoft and IBM, is the base structure for the audio files considered in this project. It provides many allowable methods for the insertion of metadata within the file. In the RIFF structure all data whether audiovisual, technical, or descriptive is contained within chunks. Standards organizations, corporations, and individuals have contributed to the development and standardization of various chunks to fulfill certain identified needs. Prior to generating a new chunk it is prudent to review existing chunks for their suitability in meeting the needs of the Working Group.

Below are some additional informative facts about the RIFF file structure and chunks:

- Chunks within a RIFF file structure include the following design:
    - a 4 byte chunk identifier (example: bext, data, aXML),
    - a 4 byte statement as to the byte size of the data within the chunk,
    - the data (whether metadata, raw audio data, or other),
    - a padding byte if the length of the data is of an odd-byte length.
- Two of the chunks RIFF and LIST are designed to contain other chunks as sub-chunks within the data section. The RIFF chunk exists as the entire file, containing all other chunks. Thus whenever the size of a RIFF's internal chunk is modified the size of the RIFF chunk must also be modified.
- A RIFF decoder is able to parse the hierarchy of chunks in a RIFF file structure by quickly scanning from the head of one chunk to another and capturing the individual chunk identifiers and their corresponding sizes. Here is an example of chunk identifiers and corresponding size data from a fairly complex audio file.

```
RIFF(WAVE)-> (6115276 Bytes)
fmt ; (16 Bytes)
bext; (858 Bytes)
FLLR; (3178 Bytes)
data; (5892096 Bytes)
filr; (88 Bytes)
filr; (59392 Bytes)
filr; (747 Bytes)
filr; (43516 Bytes)
iXML; (88 Bytes)
ID3 ; (59392 Bytes)
SNDM; (747 Bytes)
SMED; (43516 Bytes)
ovwf; (11536 Bytes)
```

Broadcast WAVE and bext

Broadcast WAVE[2] is a specific implementation of the RIFF and WAVE standard that requires the RIFF chunk to contain at least three sub-chunks: bext, fmt, and data. The format sub-chunk

declares necessary technical information about the raw audio contained in the data sub-chunk to be properly interpreted. The fmt sub-chunk expresses the encoding standard, number of channels, average bit rate, and bits per sample.

The bext sub-chunk (also known as Broadcast WAVE extension) is a 602 byte string plus an extendable value called 'coding history'. The bext chunk contains a short defined list of fields of fixed length. The data section of bext chunk contains these following values:

Byte Position	Value Name	Maximum Length (in bytes or ASCII characters)
001 - 256	description	256
257 - 288	originator	32
289 - 320	originator reference	32
321 - 330	origination date	10
331 - 338	origination time	8
339 - 342	time reference	8
347 - 348	version	2
349 - 412	UMID (according to SMPTE 330M)	64
413 - 602	reserved	190
603 - ...	coding history	extendable (often padded to 256 bytes)

Excepting 'coding history' which is able to be extended in length as needed, all fields within a bext chunk must not be longer than the length listed in the table above. If the value of that field is less than the maximum length the field is “padded” (null characters are added) so to reach the maximum length for that field. The values within a bext chunk are designed to be parsed out according to their position within the bext chunk. As a visual example, here is an ASCII and hexadecimal representation of the beginning of a sample bext chunk:

```
bext....RYL_1356, local, shelf number for source item..........................................
...................................................................................
.................................................................Library of Congress MBRS/RS
S....DLC loc.mbrsmi/westhpp.203......2005_11_3012:00:00.........................................
...................................................................................
...................................................................................
..........................A=PCM,F=96000,W=24,M=mono,T=preservation master..A=PCM,F=44100,W=16,M
=mono,T=access master; right channel; Wavelab 6.10; UV22HR dither
```

```
52494646c24d0f1157415645666d7420100000000100010044ac000088580100020010004c49535498000000494e464f4
94e414d2800000043696e6570726f62653a20416e204576656e696e67207769746820427275636520436f6e65720049
5346540e000000576176654c616220362e3130000049435244160000000382f31322f3230303820323a32383a303020504
d000049535246280000005175617274657220747261636b20312f3422204f70656e205265656c20417564696f74617065
00006265787 4e40200005 2594c5f313335362c206c6f63616c2c207368656c66206e756d6265722066 6f7220736f75726
365206974656d000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000004c696272617279206f6620436f6e677265737320 4d4252532f
52535300000000444c43206c6f632e6d6272736d692f776573746870702e32303300000000000000323030352d31312d333
031323a30303a3030b0d30f000000000000100000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000413d50434d2c463d39363030302c573d32342
c4d3d6d6f6e6f2c543d7072657365727661746696f6e206d61737465720d0a413d50434d2c463d34343130302c573d3136
2c4d3d6d6f6e6f2c543d616363657373206d61737465723b207269676874206368616e6e656c3b20576176656c6162203
62e31303b2055563232485220646974686572
```


The short and fixed length of fields in the bext chunk often places the user in a position where the data they would like to enter exceeds the limitations of a given field. The de facto common practice that has emerged for dealing with this is to place the additional information into the description field, which has a relatively large maximum of 256 characters. Thus, in addition to requiring the rules established to parse a bext chunk into the corresponding metadata values, an additional set of organization-specific rules must be developed, documented and understood by humans or machines to parse, interpret and act on the values populating the bext chunk.

LIST INFO

The LIST INFO chunk was developed by Microsoft as part of the RIFF file specification. The LIST chunk may contain a list of other chunks which each include a chunk identifier, a size, and the data. The chunk identifiers include[3]:

Sub-chunk identifiers within the LIST chunk	Field name
IARL	Archival Location
IART	Artist
ICMS	Commissioned
ICMT	Comment
ICOP	Copyright
ICRD	Date Created
ICRP	Cropped
IDIM	Dimensions
IDPI	Dots Per Inch
IENG	Engineer
IGNR	Genre
IKEY	Keywords
ILGT	Lightness
IMED	Medium
INAM	Title
IPLT	Number of Colors
IPRD	Product
ISBJ	Subject
ISFT	Software
ISHP	Sharpness
ISRC	Source
ISRF	Source Form
ITCH	Technician

The LIST INFO chunk's advantage over the bext chunk is that the supported field names are more comprehensive and the values have a variable rather than fixed length. The metadata stored in this chunk may be more precisely described or labeled than is possible in the bext chunk. Similar to the bext chunk the individual fields may not be repeated, although with LIST

INFO the less constrained length of the chunks allows for multiple values to be concatenated and listed within the chunks. The LIST INFO function also has a wide level of support across audio applications, but the level of support varies as some applications support only a limited list of potential LIST INFO chunks.

Utilization of the LIST chunk was initially documented by Microsoft in specifications regarding the RIFF file structure and multimedia software development. Although best practices exist, the standard identifiers of the sub-chunks of LIST INFO appear to be completely at the discretion of the developers and users.

iXML

The iXML chunk was 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. The iXML chunk contains a defined XML document for production information such as 'project', 'tape', 'note', 'user'[5][4]. The iXML chunk is more flexible than the bext chunk in that specifications of the chunk do not restrict the length of the metadata allowed in the chunk. Whereas the 'description' field in the bext chunk is strictly limited to 256 characters, the 'note' field in the iXML chunk is only limited by the specifications of the RIFF specification (the size of a chunk cannot exceed 4 GB when a 16 bit value describes the chunk size). The iXML chunk also supports several dozen more metadata fields than bext.

A website for iXML[5] is maintained by Gallery[6]. Although the iXML chunk is not a formal standard it has a wide implementation among professional audio recording and editing tools. A directory of software that integrates the iXML chunk is available at http://www.gallery.co.uk/ixml/compatible.html, which includes tools by Digidesign, Gallery, Apple, Nagra, and others. Implementations of the iXML chunk include disk-based field recorders, non linear video editing systems, digital audio workstations, asset, management systems, and audio playback systems.

The degree of iXML implementation varies across vendors, ranging from supporting a few fields to the entire set of fields. The iXML website hosts documentation describing what values are implemented into each supporting tool. For instance, some recording devices only write certain values to the file at the time of recording whereas an iXML chunk editor may enable fuller access to modifying an iXML document.

The ixml.info website, maintained by Gallery, offers a sample Broadcast Wave file with an embedded iXML chunk for the purpose of evaluating iXML readers. The sample file fails validation as a RIFF file, because the requirement in the RIFF specification to pad a byte at the end of any chunk of odd-byte length is not followed on the iXML chunk; therefore all subsequent chunks are one byte off causing the file to be misinterpreted and invalid. It is not known whether this is a systematic problem with software and hardware supporting iXML or if this is an aberration.

The iXML chunk is extensible by the community. The ixml.info website maintains an iXML custom registry to list new XML elements added by various organizations in order to prevent conflicting use of specific tag names.

XMP

The XMP chunk (which is identified in the RIFF file structure as a chunk using the letters ' pmx') is similar to iXML in that it is a fixed set of metadata defined and maintained by Adobe and not a standards organization. Adobe's XMP standard is highly extensible and capable of incorporating a variety of descriptive standards developed or adopted by Adobe, such as:

- Dublin Core
- XMP Basic Schema
- XMP Rights Management Schema

- XMP Media Management Schema
- XMP Basic Job Ticket Schema
- XMP Paged-Text Schema
- Adobe PDF Schema
- Photoshop Schema
- Camera Raw Schema and others

The schemas documented by Adobe allow for a wide variety of needs, but the XMP specifications define best practices in the event a user determines that Adobe's selected schemas are not sufficient. XMP has been widely integrated into a long list of Adobe and non-Adobe digital image applications and to a lesser extent into document-based application. Currently integration of XMP in audio is in the early development stages.

aXML

The aXML chunk is defined in EBU 3285 Supplement 5[7] and named after an XML expression of the Dublin Core based core audio descriptive metadata standard developed within the EBU. The specification allows for the storage of any valid XML document (version 1 or higher) that may be of any length (limited to RIFF specifications) and may appear in any order with the other chunks. Unlike the other chunks evaluated, the aXML chunk does not constrain how the user defines the data. The user may implement any XML-based metadata standard within the aXML chunk, thus the user is in more control of how their data should be defined and organized.

Although it is technically able to exist in a WAVE file alone, the aXML chunk appears to be implicitly tied to the existence of the bext chunk. This is based on the fact that the only place which aXML is specified is as a supplement to the Broadcast Wave Format. Therefore, a file must comply with Broadcast Wave standards before the aXML chunk may be applied, and thus a file containing an aXML chunk must also contain a bext chunk. This potentially allows for a

technician to utilize the bext chunk for information pertinent to the reformatting process while the aXML chunk is used to contain a related metadata document.

Another advantage to the aXML chunk is that it may appear in any order with the other chunks whereas the LIST INFO and bext chunks must be placed near the beginning of the file. The modification of chunks that appear before the data chunk make it more likely that modification of the metadata will require the entire file to be rewritten. Modification of a padded aXML chunk will only require rewriting a specific portion of the file, expediting workflows and enabling faster processing.

This document solely refers to aXML as used to identify the audio file chunk described in the Broadcast Wave Format supplement here *www.ebu.ch/CMSimages/en/tec_doc_t3285_s5_tcm6-10485.pdf*. It should be noted that the name 'axml' also refers to a profile of Dublin Core described at http://www.aes.org/e-lib/browse.cfm?elib=12826. In this document axml refers to a representation of Dublin Core metadata elements, but expanded to include group-level data, xml identifier attributes, additional implementation guidelines, and new technical vocabulary corresponding to P-Meta. Although the aXML chunk can carry any type of XML content, it is worth noting that the EBU developed an XML schema for the purpose called P_META. Version 2.0 of the specification was published in 2007 (http://www.ebu.ch/CMSimages/en/tec_doc_t3295v2-2007_tcm6-53551.pdf). The overview to the specification states that "P_META represents the 'semantic layer', also known as the 'descriptive metadata layer,' that is the exact definition and meaning of each element of description considered to be representative of common production practices." There are other EBU metadata structures, including EBU Core (based on Dublin Core), most recently revised in 2008. The Working Group is not analyzing or comparing EBU metadata schema at this time but may take up the topic in the future.

Comparative Review

This report examines chunks according to the following identified criteria:

- limitations on size: such as character limits
- limitations on definition: such as the available list of metadata fields available for user, ability to customize metadata fields or arrange them in a relational hierarchy
- level of adoption: how ubiquitous is the software support for editing and adding metadata
- the level of authority of the standard: who documented the standard and is it maintained
- extensibility of the implementation of the chunk
- structural requirements, including where within the RIFF structure it must exist. (Note: Chunks which require being positioned at the head of the file are more likely to cause a full rewriting of the file during modification and thus slow down workflow and increase opportunity for errors in rewriting the audio data.)

Chunk	Size	Definition	Adoption	Authority	Extensibility	Storage
bext	Highly Limited	Highly Limited	High	EBU[8]	None	Must be before data chunk, at the head of the file.
LIST INFO	Flexible	Limited	Somewhat High	Microsoft[9]	Unclear[10]11	Must be before data chunk, at the head of the file.
iXML	Highly Flexible	Limited, but Extensible	Moderate[11]	Collection of corporations, website maintained by Gallery	High, may be expanded as needed (registration encouraged)	May appear in any order with the other chunks of the RIFF structure
XMP	Highly Flexible	Somewhat limited, but extensible	In Development within Adobe Products.	Adobe	High, may be expanded as needed (best practices provided)	May appear in any order with the other chunks of the RIFF structure
aXML	Highly Flexible	Highly Flexible	Not commercially available. Apparent internal custom uses within organizations.	EBU[12]	Very High	May appear in any order with the other chunks of the RIFF structure, requires the file to meet BWF specifications

bext: The bext chunk is standardized within EBU and made relatively more accessible than others through a variety of tools for reading and editing; however the chunk is the most limited in size, has a very short list of fields and fixed definitions. Also the level of implementation among software varies in that some implementations selectively support the bext chunk standard by not reading or writing certain fields. Beyond prospective future use of the reserved space within the bext chunk, bext is not extendable in any way that is meaningful to the Working Group.

Overall, the advantage of the bext chunk is a strong level of support across cross-platform applications and tools; however the disadvantage is that the bext chunk provides a very limited list of metadata with tight limits as to the amount of data that may be used. These characteristics make bext a weak candidate for working with functional requirements and needs which exceed the original limited scope of the bext chunk.

LIST INFO: The support for LIST INFO is similar to bext in that there are many applications that implement the standard, but the implementation is often selective and incomplete. While widely supported, the LIST INFO chunk was not created to be extensible. Extensibility that has taken place appears to have been performed without a formal process, registry or authority.

iXML: iXML is defined by a group of corporations, not standards organizations, but is supported by a large number of hardware and software. It allows for dozens of fields in a fixed XML structure and is extensible, but is primarily designed for production rather than archival metadata.

XMP: XMP is a standard defined and maintained by Adobe, well integrated into a variety of software for images, but in the early stages of development for audio. The metadata structure of XMP is flexible, allowing for many types of field definitions and long strings of data as well as extensibility.

aXML: aXML is standardized in EBU Technical document 3285 supplement 5. The advantage of the aXML chunk is that it separates the metadata definition from the implementation. With this

chunk the user may define their own metadata standards to meet their own needs, such as Dublin Core, MODS, or PBCore and use the aXML chunk to integrate that XML document into the audio file. There are no identified commercially available tools for working with the aXML chunk, but there are custom applications that organizations have developed internally. There also seems to be interest within the community for the development of aXML tools. For instance the Audio Engineering Society Standards Committee Working Group SC-03-06 on Audio Metadata and SC-03-07 on Digital Library and Archive Systems have discussed using aXML for the embedding of emerging AES audio metadata standards. The simplicity and flexibility of the aXML chunk specification enables tools to be more easily developed and maintained compared to the other reviewed chunks.

In terms of semantic loss taking place in the migration of metadata from multiple data sources to the WAVE file, the bext chunk represents the lossiest option while the aXML chunk is the most lossless. This is based on the fact that the Federal Agencies would control the definition of an exchangeable XML standard to incorporate into the aXML chunk. Since the aXML chunk does not constrain the metadata standards (beyond the XML structure) this option would allow for the Federal Agencies to utilize a pre-existing exchange standard such as MODS, Dublin Core, or PBCore which may better support exchanges of metadata among systems.

Conclusion

The authors of this document recommend aXML as the best candidate to fully meet the requirements of the Working Group. Its extensibility, standardization, and structural flexibility make it the most suitable. In order to mitigate the prospective risks involved with aXML, we recommend making the tools developed in this project openly available, free of charge. Combined with standardization within Federal Agencies, this will provide significant impetus to other archival organizations to adopt these practices and standards.

Since aXML may contain any valid XML document, the working group may adopt or create an XML standard that fits the identified needs rather than work in the limitations of accessible chunks not designed for these purposes. The prospective use of existing XML standards such as Dublin Core or PBCore would allow the working group to store multiple identifiers, long descriptions, and a variety of other technical, descriptive and rights metadata that would be more difficult to implement within the other chunks.

As aXML is a supplement for the Broadcast Wave Format, which includes the use of bext, the authors recommend implementing a bext-based workflow prior to an implementation of an aXML-based solution. In consideration of this, the initial planning and implementation strategies should initially examine and deploy methods for the embedding of bext information prior to aXML.

Tool Assessment

A number of existing tools were examined for their ability to meet the identified needs. Tools examined included:

- WaveLab
- Audacity
- SoundForge
- Quadriga
- Dobbin
- JHOVE
- ShowRIFF
- RIFFTree
- GetID3
- MediaInfo
- WaveView

The text in this report largely speaks to the inadequacy of any one, or combination of these tools to meet the defined needs in this report. However, Dobbin deserves special note and discussion. Primarily it enables the automated creation of processing, derivative files, metadata mapping and quality reporting. Dobbin meets some of the core requirements noted in this document. It does have the capability to map metadata into the bext chunk. Theoretically, it can also be linked to a variety of data sources, such as a database for mapping and embedding metadata into audio files.

Dobbin is undoubtedly a useful tool that adds extended automation, processing and mapping capabilities beyond what other tools offer. It is a Windows-based application that works best within the context of a sophisticated production team and the resources needed to license its use. Such a team can take full advantage of the Dobbin interface, which permits workers to represent complex data processing and management workflows through the utilization of a Visio-like interface. Dobbin's strong point is in the ability to set up "back-end" automated batch processes, and therefore it purposefully leaves out a simple interface for reading, writing or editing metadata for individual files or across groups of files. This creates somewhat of a gap between more simplistic tools like those listed above and the sophistication of a tool like Dobbin. It is also worth noting that, if the Working Group does move forward with an aXML implementation, this is not supported by Dobbin at this time.

Implementation Strategies

This section examines workflows and tools necessary to utilize a RIFF chunk in order to meet Working Group goals and objectives. In examining these matters it is important to consider metadata flow, retention of metadata semantic meaning, efficiency and integrity issues.

Metadata Preparation

Embedded metadata will greatly aid in the management of audio files but is not likely considered to be the most authoritative metadata record for the asset. The authoritative record will continue to be maintained in various local databases such as MySQL, Microsoft Access, and Filemaker, or disseminated via public catalogs or finding aids. The databases, metadata standards and data dictionaries will vary greatly across the Federal Agencies. For this reason the development of a conformance point file representing an agreed upon set of fields, structure and rules is recommended. This file will function as an intermediate between the databases of the Federal Agencies and the tool used to embed metadata into the audio files.

The development of a conformance point file for a bext solution should deliver:

- a selected data exchange format such as comma or tab delimited text files representing multiple fields or multiple records
- naming convention for the conformance point file
- rules to determine the association between the conformance point file and the associated audio file

The development of a conformance point file for an aXML-based solution should add:

- an XSD that defines the metadata structure, minimal standards and rules for the conformance point file
- style-sheets or translation tools necessary to transform the database exports allowed by the systems maintaining archival metadata into the conformance point file

bext integration

Since the addition of a bext chunk into an audio file typically requires the rewriting of the entire audio file, the authors recommend that the bext chunk be introduced into the audio file as early in the reformatting process as possible. The authors found that the rewriting of the file which takes place in most audio applications after the editing of the bext chunk is unnecessary in most cases. For this reason it is further recommended that a tool be developed for reading and editing the bext chunk which takes maximum opportunity to avoid rewriting the entire file.

Batch Automation Considerations

None of the existing tools examined in this project had potential to embed data from a database (or intermediate conformance point file) into any chunks in individual WAVE files in a batch mode. The existing tools allow for individual audio files to be opened and to edit the LIST INFO or bext metadata one-at-a-time. This workflow is burdensome, allows a greater potential for errors, proceeds at a much slower rate, and encourages the entry of this metadata to occur at a point in the workflow when it may not be most efficient.

An automated tool for bext chunk creation and editing should meet the following requirement:

- test for a pre-existence bext chunk,
- allow for batch editing of selected bext fields,
- validate bext fields that require formatting such as timeReference, originatorDate, and UMID,
- properly adjust chunk size references as needed,
- validate the edited results as a valid to RIFF and BWF specifications.

A tool capable of adding an aXML chunk and metadata to a RIFF file in a batch mode should meet the following requirements:

- validating the source audio file as a Broadcast Wave Format file
- validating the source data as valid XML
- optionally validating the XML against an XSD
- appending the corresponding aXML identifier, the size of the XML document, and the XML data to the tail of the file
- adjusting the file size noted in the RIFF chunk at the very beginning of the file

With these requirements met it is anticipated that batch automation is possible, overcoming the aforementioned issues.

File Rewrites

File rewrites are a significant detail to address, as they can represent a bottleneck in the workflow, discourage editing of metadata, and increase the likelihood for error and loss. Editing of metadata often changes the length of the chunk in which it resides, causing all subsequent data to require a rewrite in order to offset it accordingly. Since most of the bext chunk is a fixed length, the editing of a bext chunk (excepting the variable length coding History field) should not technically require that the entire file be rewritten but only the few revised bytes within the bext chunk.

In review of existing audio processing tools, all tools examined required re-writing the entire audio files even when the modification of metadata did not offset the preexisting data of the file. As tools are developed in this project the authors recommend that unnecessary file rewriting be avoided. In a batch process this would slow down the application dramatically. There are two strategies to prevent file rewrites during metadata modification:

- Pad the file with filler chunks. A filler chunk consists of the identifier fllr with a size, then a data section filled with zeros. With this method the chunk preceding the fllr chunk may then be expanded as needed to overwrite the fllr chunk and then a new fllr chunk

header would need to be rewritten to contain the unused bytes of the partially overwritten fllr chunk. This option is used internally in a few software applications, but its integration in this workflow would require that filler chunks be introduced early into the reformatting process which would be difficult to support.

- Insert the metadata chunk at the tail of the file. With this option the new metadata may be appended to the end of the file. None of the pre-existing data within the file would require a rewrite (with the exception of modification of the size expression in the RIFF chunk); however this option would require using a metadata option that can be written after the data chunk. Both bext and LIST INFO must be written near the beginning of the file thus are more likely to require file rewrites upon modification whereas aXML, iXML, and XMP chunks may occur anywhere within the file and thus may be appended to the end.

Since file-rewriting can slow the workflow considerably and even discourage the correction of metadata there are two recommendations:

- Develop a tool that takes advantage of previously unleveraged bext chunk allowances for adding/editing metadata while avoiding file rewrites. This tool would effectively allow for the modification of all bext values with negligible file rewrite times while restricting the users ability to extend the Coding History value beyond what is already allocated to that field
- Develop a tool to allow xml documents to be appended to the end of a file as an aXML chunk

Checksums

Modifications of the archived file by means of inserting additional metadata will adjust the checksum of the file and negatively impact workflows relying on the use of the checksum's hash

value. Potentially a chunk like aXML is comprehensive enough to allow an aXML tool to retrieve and store versioning information about the checksum of the audio file within the chunk; however it would be difficult to control and document all such modifications with a checksum history.

Since the archived file may undergo occasional modifications as embedded metadata is updated individually by technicians or ongoing by a batch-processing metadata insertion tool, it is recommended to retain a checksum that documents the data chunk of the audio file specifically. This will effectively create a checksum hash value for the audio portion of the file only, avoiding changes to the hash value when embedded metadata values are altered. This can serve as a supplement to the hash value for the file as a whole to provide comprehensive integrity monitoring.

Metadata Workflow and Tool Design

In order to accommodate various workflows the development of a metadata tool(s) should meet the following requirements:

- operation in singular or batch mode
- validation of an input conformance point document as valid according to either the bext specifications or to XML specifications if aXML
- validation of the input audio file as valid Broadcast Wave Format
- avoidance of unnecessary file-rewriting
- enable individual editing of the embedded bext fields, in batch mode of singularly
- the ability to extract XML documents from the XML-based chunks

Summary

In summary, the value of embedding metadata within files (WAVE files in this case) is agreed upon by the members of the Working Group. With this in mind the Working Group is in the process of documenting the list of fields and the associated data dictionary. This will make up the core metadata set that they would like to embed in WAVE files. The two primary candidates for embedding metadata within WAVE files currently are the LIST INFO chunk and the bext chunk. It is apparent when the field list and data dictionary are compared to these chunks that they are not sufficient with regard to the fields they offer or their flexibility. However, their adoption within the marketplace makes them attractive. Investigation of other metadata embedding options shows that the aXML chunk most closely aligns with the requirements and goals of the Working Group's use of the core metadata set.

Because embedding is considered as valuable and necessary, and WAVE files are being produced in great numbers within the Federal Agencies it is imperative that embedding begin immediately despite the lack of a readily available ideal solution. The bext chunk is recommended to fill this role for immediate implementation of embedding the Working Group's defined core metadata set.

In the meantime it is recommended that simultaneous work take place toward developing tools that will utilize the aXML chunk for embedding of the Working Group's core metadata set.

Endnotes

[1] http://msdn.microsoft.com/en-us/library/ms713231.aspx

[2] see EBU tech doc 3285 http://www.ebu.ch/CMSimages/en/tec_doc_t3285_tcm6-10544.pdf

[3] A more comprehensive list supportive within AVI files is available at http://abcavi.kibi.ru/infotags.htm

[4] see full example of the iXML implementation at http://www.gallery.co.uk/ixml/iXML_Example.html

[5] http://www.ixml.info

[6] http://www.gallery.co.uk/

[7] available at http://www.ebu.ch/CMSimages/en/tec_doc_t3285_s5_tcm6-10485.pdf

[8] See Specification of the Broadcast Wave Form Technical Document #3285

[9] The LIST INFO chunk appears to be originally documented by Microsoft and IBM in defining the RIFF file structure. However there is no seeming active authority managing the LIST INFO chunk.

[10] Extensibility has been practiced but appears to be informal and without an authority or formal registry.

[11] Adoptions of iXML are selective in their support, see http://www.gallery.co.uk/ixml/compatible.html

[12] See Specification of the Broadcast Wave Form Technical Document #3285, Supplement 5: <axml> chunk http://www.ebu.ch/CMSimages/en/tec_doc_t3285_s5_tcm6-10485.pdf