PROPOSED SMPTE STANDARD

SMPTE 382M

for Television — Material Exchange Format (MXF) — Mapping AES3 and Broadcast Wave Audio into the MXF Generic Container

1 Scope

This standard defines the mapping of digital audio data, ancillary data and metadata from the broadcast wave format (BWF) and from densely-packed AES3 digital audio data into sound essence elements , which can be included in an MXF generic container. The data carried in wave audio essence elements may be uncompressed PCM audio data, compressed data or raw data as in BWF, AES3 and SMPTE 337M. Labels for the identification of uncompressed audio samples within this mapping are provided by this standard. Labels for compressed audio types can be added to SMPTE RP 224.

In order to achieve interoperability within any given operational pattern, restrictions may be placed on the way in which this essence container can be implemented. the reader is advised to carefully study the appropriate operational pattern document before implementation.

2 Normative references

The following standards contain provisions which, through reference in this text, constitute provisions of this standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreement based on this standard are encouraged to investigate the possibility of applying the most recent edition of the standards indicated below.

SMPTE 337M-2000, Television — Format for Non-PCM Audio and Data in an AES3 Serial Digital Audio Interface

SMPTE 338M-2000, Television — Format for Non-PCM Audio and Data in AES3 — Data Types

SMPTE 339M-2000, Television — Format for Non-PCM Audio and Data in AES3 — Generic Data Types

SMPTE 377M-2004, Television — Material Exchange Format (MXF) — File Format Specification

SMPTE 379M-2004, Television — Material Exchange Format — MXF Generic Container

SMPTE RP 224, SMPTE Labels Registry

AES-3 (2003), Serial Transmission Format for Two-Channel Linearly Represented Digital Audio Data

EBU Tech 3285-2001, Specification of the Broadcast Wave Format — A Format for Audio Data Files in Broadcasting: Version 1

EBU Tech 3285 Supplement 2-2001, Specification of the Broadcast Wave Format — A Format for Audio Data Files in Broadcasting — Supplement 2: Capturing Report (Specification of the Quality Chunk)

EBU Tech 3285 Supplement 3-2001, Specification of the Broadcast Wave Format — A Format for Audio Data Files in Broadcasting — Supplement 3: Specification of the Peak Envelope Chunk

ITU-R BR.1352-2:2002, Broadcast Wave Format (BWF), Annex 1, Annex 1 Appendix 1 and 2, and Annex 3

NOTE – The MXF mapping provides for the mapping of audio samples into MXF content packages and the mapping of broadcast wave chunk information into MXF metadata. This standard defines mappings appropriate to both EBU and ITU standards, and efforts are in progress to remove any difference between the ITU and EBU versions of broadcast wave. This work is being carried out in the Audio Engineering Society Standards Committee, with the goal of harmonizing the various broadcast wave variants. Extension mechanisms are defined in this standard for the transparent carriage of broadcast wave essence and metadata in MXF. This allows MXF systems that support this standard's mechanisms to transparently carry and recreate the audio data and metadata in a broadcast wave file.

3 Glossary of acronyms, terms and data types

Explanations and definitions of the acronyms and terms can be found in the MXF file format specification document. A supplementary glossary of acronyms and terms is defined in SMPTE 379M. They are not repeated here to avoid any divergence of meaning.

The following acronyms are used in this standard:

• AES	Audio Engineering Society
• BEXT	The Broadcast Extension RIFF Chunk
• BWF	Broadcast Wave Format defined in EBU Tech T3285-2001
• CRCC	Cyclic Redundancy Check Code
• LEVL	The Peak Envelope (level) RIFF Chunk
• PCM	Pulse Code Modulation
• RIFF	Resource Interchange File Format — a generic file format consisting of a series of chunks, each composed of a 32-bit chunk identifier, a 32-bit length field, and a number of data bytes.
• QLTY	The Capturing Report (quality) RIFF Chunk

The following additional terms apply to BWF and AES3. Refer to AES3 for the precise definition.

Subframe: a single AES3 data frame, 32 bits including preamble, aux, sample data, V U C and parity bits.

Frame: two subframes, for channels 1 and 2.

Block: a group of 192 consecutive frames. The start of a block is designated by a special subframe preamble.

The following additional data types apply to this standard:

char[ ] A data type comprising either a single ISO 7-bit character or a string of ISO 7-bit characters with the string length determined by the number in the brackets. Where brackets are specified as simply "[ ]", the string length is variable and terminated by a zero value.

In some geographic areas, strings of type char[ ] may contain UTF-8 or Shift-JIS characters.

4 Introduction

This standard defines the mapping of digital audio data and ancillary data and metadata from broadcast wave format (BWF) files, and densely-packed AES3 data streams into KLV-encoded sets for inclusion in MXF files.

In common with other body specifications, the principal mapping is of the essence, in this case samples of digital audio data, into KLV packets. These are called wave audio essence elements.

This standard employs the same bit-by-bit packing of sample data as is used by BWF. The sample data is packed on a frame-by-frame or clip-by-clip or 5-frame basis excluding all ancillary data. This is described in section 5.

This standard defines wave audio essence elements which may be used as MXF essence containers in their own right or as elements in content packages within an MXF generic container, depending upon application requirements. The division of audio essence into elements is discussed in section 5.2.

Data carried in wave audio essence elements may be uncompressed PCM audio data or compressed data, as defined in BWF, AES3 and SMPTE 337M. The specific format is indicated by the SoundEssenceCoding property, which is described in section 6.2.1.

BWF defines metadata which applies to the entire clip. This standard maps this metadata into the MXF header metadata. This is described in section 6.

This standard also describes the mapping of constant and time-varying ancillary data and metadata from AES3 subframes and U and C bits into appropriate MXF data structures. This is described in section 8.

NOTE – Data may be lost if applications do not support the carriage of the broadcast wave metadata or aes metadata parameters which are not mapped to the MXF structural metadata. Implementers and users should ensure that descriptive metadata mappings or other mechanisms are available in applications which require this level of data integrity. This particularly applies to parameters such as unmapped BWF chunk parameters as described in section 6.3.

5 Packing of audio samples

This standard defines audio essence elements containing one, two or N channels.

In a multiple-channel element, samples are interleaved in order of channel number. The top-level file package shall have one track per mapped wave audio essence stream.

A material package may have multiple tracks of type sound. SourceClips in the material package may reference a given channel or channels within the top-level file package by setting an optional ChannelID property to the channel number in the target track.

When AES3 subframes are being mapped, the first subframe of a block (i.e., that identified by a Z preamble) is first in every sample pair.

All ancillary data, including the VUCP bits, are mapped to a wave audio essence element as part of the packing of audio samples. Section 5.1 considers the validity bit, while section 8 describes the handling of the user, channel status and parity data.

Each sample shall be carried in the smallest number of bytes necessary to completely contain the data, stored as a little-endian integer (least significant byte first). Data shall be aligned to the most significant bit, and unused bits shall be set to zero.

For example, 20 bit samples are carried in 3 bytes. The least significant 4 bits of the first (least significant) byte are set to zero. A decoder can determine the number of bytes from the essence descriptor using the formula given in section 6.2.

NOTE – EBU Tech 3285 and its supplements provide additional details of the mapping of audio samples into data bytes. SMPTE 320M and SMPTE 323M provide definitions of commonly used assignments of multiple channels. ITU-R BR.1352 may also have details of the mapping of audio samples into data bytes.

This standard does not define the format of MXF essence containers or essence descriptors for big-endian sound essence.

5.1 Sample validity

When formatted according to AES3, audio subframes include a channel validity (V) bit, which is set to one when the sample data is not suitable for conversion to an analog audio signal.

This data is not available in the packed wave audio essence elements data format, and all samples are presumed to be valid.

NOTE – SMPTE 314M section 4.6.1.2.3 describes an audio error code in DV-derived audio sample data. This error code is the most-negative value of the sample (8000h), and is used to indicate invalid samples. These samples should be overwritten with valid data in the data stream before packing into wave audio essence elements. This applies only to DV-derived audio; for AES-3 derived audio, 8000h is valid sample data. DV-derived audio sample data can be determined from the SoundEssenceCoding parameter which may be extended using the SMPTE Labels Registry (SMPTE RP 224) to include DV-derived audio types.

5.2 Division of audio essence into frame wrapped content

Audio essence may be divided into elements of approximately equal duration, for synchronizing with video essence within content packages of an MXF generic container. “Approximately equal duration” is defined as a constant number of samples ± 5 samples.

The division into elements can take a variety of forms which are driven by various system design issues. This section describes four example divisions drawn from current practice:

- Audio elements each as long as the corresponding picture frame.
- Audio elements where the duration of the elements in each content package matches the duration of the synchronized video within that content package.
- Audio elements where each has the same duration as some fixed number of video frames. Generally, the duration (or number of frames) is chosen to reduce or eliminate the variation in the size of each group, or is chosen to be convenient for buffering considerations.
- Audio elements divided into content packages where the duration of those elements is convenient for buffering, such as one or two seconds.

In the case of audio elements with a duration corresponding to one video frame, the size of each element may or may not vary in size, depending on the numerical relationship between the audio sample rate and the video frame rate. For example, with video having a rate of 25 content packages per second, and audio having a sample rate of 48 kHz, each audio element will contain exactly 1920 samples.

However, in the case of audio locked to video having a rate of 30*1000/1001 (≈ 29.97) content packages per second and divided into elements approximately corresponding to a video frame, the number of samples in each element will vary in order to maintain a correct aggregate rate (about 1601.6 samples per video frame) This can be approximated with a 5-frame pattern, for example 1602, 1601, 1602, 1601, 1602, or 1600, 1602, 1602, 1602, 1602; other sequences are possible. The ordinal number of the first frame of audio essence within a five-frame sequence may optionally be indicated in the SequenceOffset property of the wave audio essence descriptor (see annex A.1), to provide guidance in processing. Note that in this instance, there will be a variable number of samples per video frame which may require the audio samples to be indexed as variable rate elements.

In the case of there being no interleaved picture content, then one frame, for the purposes of frame wrapping, shall be equivalent to the definition of the edit unit in the top-level file package track which is linked to the sound element keys via the mechanism defined in SMPTE 379M. It is recommended, however, that clip wrapping be used when there is no interleaved picture content.

The number of samples in each content package is calculated from the length field of the surrounding KLV packet, divided by the value of the BlockAlign property of the wave audio essence descriptor.

Two content packages, each containing two sound elements, are shown in figure 1.

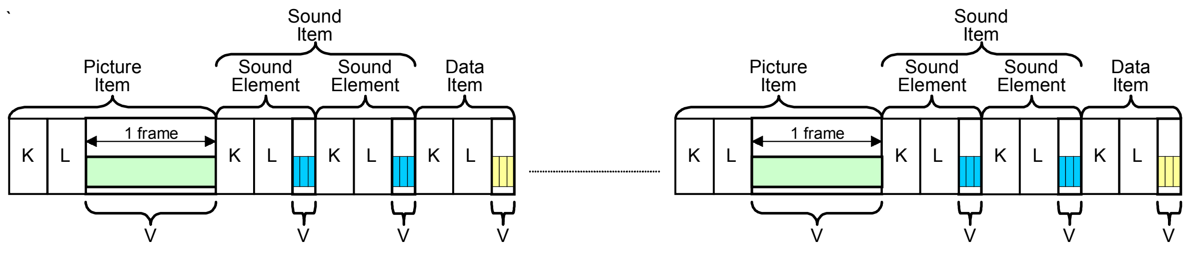

Figure 1 – Representation of frame wrapping with other elements present

The generic container key for the sound item is given in section 5.5.1.

5.3 Division of audio essence into clip wrapped content packages

An alternative to frame wrapping is to include all the audio data in a single element. This is defined as clip-based mapping in SMPTE 379M.

This wrapping may also be useful in sound-only applications.

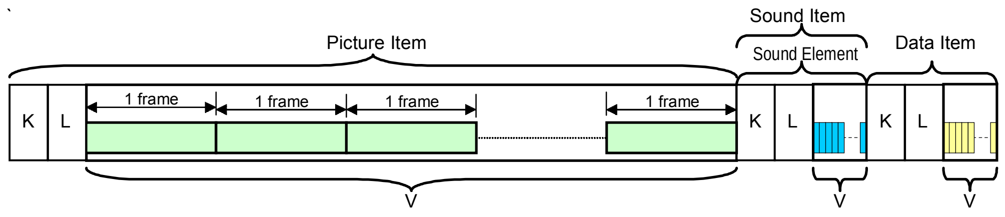

Figure 2 – Simple representation of clip wrapping with other elements

5.4 Custom wrapped audio elements

There may also be the case where a logical pattern of video frames occurs. For example, with MPEG-2, the shape of the GOP (group of pictures) may not be constant, and it may be desirable for the arrangement of audio elements within a content package to correspond to each GOP. For example, this first GOP may be four frames in length (with frame types of IPBB in transmission order), while the second GOP may be ten frames in length (IPBBPBBPBB). The length of audio elements could be chosen in lengths approximately equal to these logical frame groups.

There may be other cases when the audio wrapping is slaved to some external criteria where the “approximately constant” number of samples per element cannot be guaranteed. This use case shall be treated as a custom wrapped audio element

When audio elements are created where the duration rules of frame wrapping and clip wrapping are not met, the custom wrapping mode shall be used.

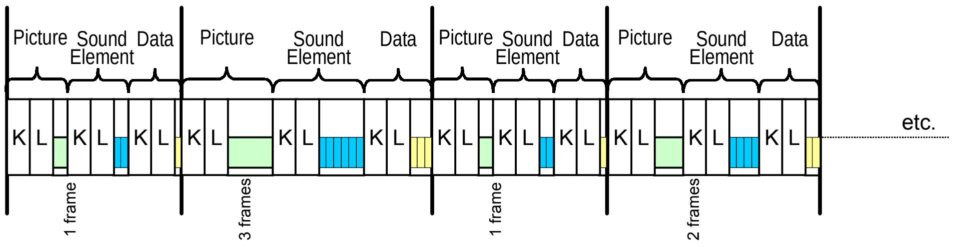

Figure 3 – Simple representation of custom wrapping with other elements

5.5 KLV coding of digital audio data sound elements

5.5.1 Wave sound element key

The values of the first 12 bytes of the essence element key are defined in SMPTE 379M, MXF generic container format. The values of the last four bytes of the essence element key are given in table 1:

Table 1 – Key value for the wave sound element

Byte No.	Description	Value (hex)	Meaning
13	Item Type Identifier	16h	GC Sound Item
14	Essence Element Count	kkh	Count of Sound Elements in this Generic Container
15	Essence Element Type	01h 02h 03h 04h 0Bh 0Ch	Wave Frame-Wrapped Element Wave Clip-Wrapped Element AES Frame-Wrapped Element AES Clip-Wrapped Element Wave Custom-Wrapped Element AES Custom-Wrapped Element
16	Essence Element Number	nnh	The Number (used as an Index) of this Sound Element in this Generic Container

5.5.2 Essence element count — Byte 14

This is a count of the number of elements in the sound items in the generic container.

5.5.3 Essence element type — Byte 15

The values are given in table 1.

5.5.4 Essence element number — Byte 16

This is a number used as an index to identify this instance of the element type within the item. Each element within an Item shall have a unique value between 0 and 7Fh as defined by SMPTE 379M, which shall remain constant throughout any instance of a generic container.

5.5.5 Wave sound element length

The length field of the KLV coded element shall be 4 bytes BER long-form encoded (i.e., 83h.xx.yy.zz) for frame wrapping. The length field of the KLV coded element shall be 8 bytes BER long-form encoded (i.e., 87h.aa.bb.cc.dd.ee.ff.gg) for clip wrapping.

5.5.6 Wave sound element value

In a clip-wrapped file, the element value shall be the complete wave audio data.

In a frame-wrapped file, the start position of the first sample of wave audio data should be the same as the start position of the synchronized picture frame. When equal start position values cannot be achieved because of complex clocking relationships, the start position of the video access unit should fall within the duration of the first sample or audio frame of the sound element. More information on mapping to achieve synchronization is given in SMPTE EG 41, the MXF engineering guideline.

5.6 Use of KAG

There are no specific KAG requirements for the wave mapping. MXF encoders and decoders shall comply with the KAG rules in the MXF format document.

5.7 Optional mapping of material package tracks to audio channel IDs

The optional ChannelID property shall be added to source clips in a material package to select one or two channels out of N-channel essence described by a top-level file package. This is typically used to break out a single track of MultiChannel sound into individual mono sound tracks, or to identify a stereo sound pair within a multi-channel essence container.

A value of N in the channel ID property identifies the Nth channel within the essence. The first channel has a channel ID of 1.

In the case of sound compressed and mapped into AES pairs according to SMPTE 337M, it is necessary to identify the source channel in greater detail, since each AES pair may contain multiple 337M streams, each containing multiple subchannels. In this case, the ChannelID shall be interpreted as a structured number according to the following bit mapping:

Bit numbers	Name	Meaning
31-24	Subchannel	1-based index of the Subchannel number within the SMPTE337M AES pair 0 implies all Subchannels in a Stream
23	StreamFlag	1 if a specific Stream is selected by bits 22-16 0 implies all Streams in a SMPTE 337M AES Pair
22-16	Stream	When bit 23=1, 0-based Stream number within the SMPTE 337M AES pair Otherwise, 0
15-0	Subframe	1-based index of the first subframe of a SMPTE337M AES pair in the Essence

NOTE – This encoding of bits 31-16 permits values of ChannelID in the range 1-65535 to have the same effect irrespective of whether the essence is compressed per SMPTE 337M or not.

The optional MonoSourceTrackIDs property shall be added to SourceClips in a material package or a file package to identify the single channel tracks of lower-level source packages which were combined to form a multi-channel track. A SourceClip shall not simultaneously contain both a ChannelID property and a MonoSourceTrackIDs property. When the MonoSourceTrackIDs property is present, the value of the SourceTrackID property of the SourceClip shall be ignored. The MonoSourceTrackIDs property is typically used on a Toplevel File Package to describe the transfer of several individual single channel inputs from a source device into a multichannel file.

Note that these properties are not intended to be used for a generalized N-channel to M-channel mapping mechanism. That functionality is best performed by splitting up the channels into discrete mono–essence essence containers and then describing them with several top-level file packages. The standard material package SourceClip mechanism can then be used to map them.

NOTES

1 These properties apply to all SourceClips (sound), not just AES/BWF SourceClips (sound). They will be proposed as an amendment to SMPTE 377M when a procedure exists for doing so.

2 In accordance with all MXF sets, the version number does not need to be incremented; all sets can be extended with optional properties. Adding required properties would require a new ID

3 In the AAF domain, this latter approach allows matrixing with AAF mixdown effects.

Table 2 – Optional properties in a SourceClip (sound)

Item Name	Type	Len	Local Tag	UL Designator	Req ?	Meaning	Default
All items from SMPTE 377M annex B.13 SourceClip (Sound)							
ChannelIDs	Array of Uint32	8+ 4n	1103	06.01.01.03.07	Opt	Ordered Array of ChannelIDs referenced by this SourceClip. [**RP210** Specifies an ordered array of channels within the referenced precursor]	
MonoSource TrackIDs	Array of Uint32	8+ 4n	1104	06.01.01.03.08	Opt	Ordered Array of Track IDs referenced by this SourceClip. [**RP210** Specifies an ordered array of tracks within the referenced precursor]	

6 Carriage of BWF metadata chunks

6.1 General

BWF defines metadata which applies to the entire clip. This standard maps this metadata into the MXF header metadata in properties of the wave audio essence descriptor (see annex A).

In BWF, clip-oriented metadata is usually stored at the head of the file in RIFF data structures (RIFF is defined in the normative reference of BWF). Defined chunks include the <fmt> chunk and the broadcast wave extension <bext> chunk. Other RIFF chunks are allowed.

NOTE – Some metadata chunks can also be stored after the audio data. According to the normative references, the only chunk that is required to appear before the audio data is the <fmt> chunk.

6.2 <fmt> Chunk

The <fmt> chunk as defined by BWF includes several parameters describing the audio essence, as follows:

```
typedef struct {
   UInt16 wFormatTag;
   UInt16 nChannels;
   UInt32 nSamplesPerSec;
   UInt32 nAvgBytesPerSec;
   UInt16 nBlockAlign;
   UInt16 wBitsPerSample;
} fmt-ck;
```

These parameters are defined in BWF section A.1.1 plus its normative reference documents.

wFormatTag	Defines the audio waveform type of the audio stream.
nChannels	Specifies the number of channels in the audio stream, 1 for mono, 2 for stereo.

nSamplesPerSec	Specifies the frequency of the sample rate of the audio stream in samples/second (Hz). Examples are 48000, 44100, and 96000.
nAvgBytesPerSec	Specifies the average data rate. Playback processors can estimate the buffer size by using this value. For VBR audio, Index Tables must be used to further refine any value in AvgBps.

nBlockAlign Specifies the block alignment of the data, in bytes. Playback processors must process a multiple of **nBlockAlign** bytes of data at a time, so that the value of **nBlockAlign** can be used for buffer alignment. Examples are: 2 (for 16 bit mono), 6 (for 20 and 24 bit dual channel).

For uncompressed PCM:

$$BlockAlign = ChannelCount \times floor\left(\frac{QuantisationBits + 7}{8}\right)$$

and the reverse is true:

$$ChannelCount = \frac{BlockAlign}{floor\left(\frac{QuantisationBits + 7}{8}\right)}$$

For compressed signals, the equations above are not valid. ChannelCount and BlockAlign are independent. ChannelCount must be specified equal to the number of channels in the decompressed signal.

For 337M, the number of AES channels (aka AES subframes) present in the essence data is

$$SubframeCount = \frac{BlockAlign}{floor\left(\frac{QuantisationBits + 7}{8}\right)}$$

Applications shall use this formula - there is no need to carry this as a separate property in the stream.

wBitsPerSample Specifies the number of bits per sample per channel data. Each channel is assumed to have the same sample resolution. If this field is not needed, it shall be set to zero.

For MXF, the parameters below are carried as individual items in the wave audio essence descriptor set. The correspondence is as follows:

Field in <fmt> chunk	MXF Set :: Property
wFormatTag	WaveAudioEssenceDescriptor::Sound Essence Coding
nChannels	WaveAudioEssenceDescriptor::ChannelCount
nSamplesPerSec	WaveAudioEssenceDescriptor::SampleRate
nAvgBytesPerSec	WaveAudioEssenceDescriptor::AvgBps
nBlockAlign	WaveAudioEssenceDescriptor::BlockAlign
wBitsPerSample	WaveAudioEssenceDescriptor::QuantizationBits

6.2.1 Sound essence compression

The sound essence compression property of the WaveAudioEssenceDescriptor in MXF is a 16-byte SMPTE UL . The equivalent field in the <fmt> chunk is wFormatTag. When uncompressed (PCM) audio is in the file, this optional property shall be absent from the set.

Other sound essence compressions (for example, MPEG 1 level 1, DV) shall be added utilizing the SMPTE Labels Registry (SMPTE RP 224). These may be drawn from the repertoire of coding formats specified in BWF, SMPTE 338M, or other standards.

6.3 Broadcast extension <bext> chunk

The <bext> chunk defined by BWF carries the following metadata:

```
typedef struct {
   char Description[256];
   char Originator[32];
   char OriginatorReference[32];
   char OriginationDate[10];
   char OriginationTime[8];
   UInt32 TimeReferenceLow;
   UInt32 TimeReferenceHigh;
   UInt16 Version;
   UInt8 UMID[64];
   UInt8 Reserved[190];
   char CodingHistory[];
} bext-ck;
```

These parameters are defined in BWF section 1.3 and companion documents.

Description	is a free text description of the sound sequence. BWF defines the coding as an ISO 7 bit character string.
Originator	is the name of the originator as an ISO 7 bit character string.
OriginatorReference	is an unambiguous reference to the material provided by the originator as an ISO 7 bit character string. See also EBU R99-1999.
OriginationDate	is the date of creation specified as an ISO 7 bit character string: “yyyy-mm-dd”.
OriginationTime	is the time of creation specified as an ISO 7 bit character string “hh-mm-ss”.
TimeReferenceLow	is the least significant 32 bits of the starting time, expressed as a count of samples since midnight.
TimeReferenceHigh	is the most significant 32 bits of the starting time, expressed as a count of samples since midnight. See BWF A.1.3.1.
Version	is the version number of the BWF format, currently defined as 0001h.
UMID	is the material UMID per SMPTE 330M. If only the basic UMID is used, the last 32 bytes shall be set to zero.
Reserved	is the specified number of unused bytes.
CodingHistory	is a text description of the coding history of the sound sequence as a null terminated ISO 7 bit character string. See BWF A.1.3.1.

Where the <bext> chunk is present, the required parameters shall be mapped exactly to items within the file package set. The remaining optional parameters (description, origination and coding history) may be mapped into a descriptive metadata scheme, for example MXF descriptive metadata scheme 1. The correspondence is shown in table 3.

Note that the first 32 bytes of the UMID (i.e., the basic UMID) are mapped into the MXF source package PackageID property to identify the essence. The remaining 32 bytes of the extended UMID may be carried according to any appropriate metadata scheme which supports this functionality. The processing and use of UMIDs is discussed in more detail in SMPTE RP 205.

Table 3 – <bext> Chunk to MXF mapping

.

Status of mapping	Field in <bext> chunk	MXF Set :: Property
opt	**Description**	ClipFramework::Annotation::AnnotationDescription
opt	**Originator**	ClipFramework::ContactsList::Person or ClipFramework::ContactsList::Organization
opt	**OriginatorReference**	ClipFramework::ClipNumber
req	**OriginationDate**	FilePackage::PackageCreationDate
req	**OriginationTime**	FilePackage::PackageCreationDate
req	**TimeReference(Low/High)**	FilePackage::AudioTrack:: Origin **NOTE –** The value stored is the negative value of TimeReference so that the sign of the Origin parameter is correct according to SMPTE 377M
	Version	(unused)
req	**UMID[0 – 31]**	SourcePackage::PackageUID
	Reserved	(unused)
req	**CodingHistory**	PhysicalPackage:: WaveAudioPhysicalDescriptor::CodingHistory

6.4 Other chunks

Other RIFF chunks may be present in a BWF file.

6.4.1 Peak envelope <levl> chunk

The <levl> chunk as described in EBU Tech 3285 Supplement 3 contains a peak envelope for the audio essence. The <levl> chunk is carried as optional properties of the wave audio essence descriptor set (see annex A.1). MXF encoders shall copy all parameters to the appropriate parts of the MXF header metadata. When recreating a BWF file, MXF decoders shall reconstruct the <levl> chunk, or shall retransmit the original chunk.

As noted in EBU Tech 3285 Supplement 3, all data in the <levl> chunk is stored little-endian in the PeakEnvelopeData property of the wave audio essence descriptor. Since this chunk contains an essence-like waveform, the data in this chunk is NOT reordered to the big-endian format of MXF metadata. The contents of this chunk shall be treated as a string of bytes.

6.4.2 Quality <qlty> chunk

The <qlty> chunk is described in EBU Tech 3285 Supplement 2-2000 and is carried as properties of the optional WaveAudioPhysicalDescriptor.

When receiving a BWF file, MXF encoders shall preserve the coding history field of the <bext> chunk and all the fields of the <qlty> chunk as individual properties in the WaveAudioPhysicalDescriptor in a physical package. Encoders may create properties in the descriptive metadata from the <bext> and <qlty> chunks according to the translations in the table in annex A.3. If a <bext> or <qlty> chunk is not present in the input file, MXF encoders may create properties in the descriptive metadata directly, and shall not create a WaveAudioPhysicalDescriptor.

When encoding audio essence directly to an MXF file, MXF encoders may create properties in the descriptive metadata directly, and shall not create a WaveAudioPhysicalDescriptor.

When creating a BWF file from an MXF file, MXF decoders should create new <bext> and <qlty> chunks from properties of the descriptive metadata. If these properties are not present, decoders may recreate the <bext> and <qlty> chunks from individual properties in the WaveAudioPhysicalDescriptor, if present.

NOTE – For compatibility with all other string properties in MXF, all properties of the WaveAudioPhysicalDescriptor are of type UTF-16 string. The corresponding fields in the BWF <bext> and <qlty> chunk are of type ISO 7 string, which may be further encoded as UTF-8 or shift-JIS in some geographic regions. When the WaveAudioPhysicalDescriptor is used to create BWF files, MXF decoders should restrict the values of these properties to those permitted in the particular region.

6.4.3 Other chunks

For MXF, these chunks are treated as dark metadata and are carried as optional properties of the WaveAudioPhysicalDescriptor Set. MXF decoders should not delete optional properties carrying BWF chunks as dark metadata. When recreating a BWF file, MXF decoders should reconstruct or retransmit these chunks.

Example: To insert the <fact-ck> chunk property called DWORD dwHeadBitrate into the WaveAudioPhysicalDescriptor, a data type would be registered in SMPTE RP 210 and would have a SMPTE UL corresponding to the property dwHeadBitrate having the data type DWORD. The primer pack mechanism in SMPTE 377M would be used to allocate this property a 2-byte dynamic tag which would then be used to insert the data value into the WaveAudioPhysicalDescriptor. An encoder or decoder wishing to use this property must know its SMPTE RP 210 UL. No further addition to this document is required to continue this extension mechanism; applications should use the latest versions of the SMPTE dictionary to discover property additions.

6.4.4 Unknown chunks while MXF encoding

An MXF encoder should carry unknown BWF chunks in unknown chunk sets (annex A.4). If one or more unknown chunks are encountered by an MXF encoder, then those chunks should each be stored as an unknown chunk set within the header metadata of the MXF file. The WaveAudioPhysicalDescriptor property "UnknownBWFChunks" shall contain a strong reference to the Instance UID of all unknown chunk sets in the file that are associated with this essence.

This mechanism enables transparent, lossless carriage of BWF metadata in MXF systems.

6.4.5 Use of a physical descriptor (informative)

Why is a physical descriptor used to hold the AES / BWF source properties and not a file descriptor? This is because the source reference chain of the AAF – MXF data model is used to contain metadata which give an audit trail of what happened to the file in the AAF – MXF environment. This linkage between source packages is essentially an AAF – MXF centric view of the world. When content enters into the MXF environment from a non AAF-MXF source, e.g. a physical tape, a physical file on disc such as BWF, AVI or other external formats, a PhysicalDescriptor is used to represent the “end point” of the source reference chain. Although it could be argued that a BWF file should be described with a FileDescriptor because it was once a file, this would not fit with the data model which demands non-MXF content be described at the lowest source level with a PhysicalDescriptor.

7 Carriage of SMPTE 339M time stamp, V-sync and user data

SMPTE 339M defines a time stamp data burst, including SMPTE 12M time code and SMPTE 309M time and date information, plus timing offset information.

In a SMPTE 337M signal, time stamp data is carried as data_type=2 in data_stream_number=7.

In the MXF header metadata, time stamp data shall be carried in the timecode sets of a time code track of the MXF file package and is not included within the MXF essence container. An MXF file package shall include a time code track for the time stamp data of each time-stamped SMPTE 337M data stream.

The additional timing precision provided by word 4 sample_count and word 8 reference_position of the SMPTE 339M time stamp data burst shall be carried in the offset property of the sound track of the MXF file package. The value of the offset property shall be the sum of word 4 and word 8.

SMPTE 339M also defines V-sync bursts, which are intended to allow identification of an alignment point between an AES3 stream containing SMPTE 337M formatted data and a corresponding video raster.

In a SMPTE 337M signal, V-sync bursts are carried as data_type=26 in data_stream_number other than 7.

In an MXF essence container, V-sync bursts shall be taken as the reference point for the start of edit units in the creation of MXF index tables and shall not be deleted from the essence data.

The spacing of V-sync bursts within the essence stream may be used to derive the SequenceOffset property of the wave audio essence descriptor.

SMPTE 339M also defines user data bursts, which are carried in SMPTE 337M data streams identified by data_stream_number other than 7. Such streams shall not be carried by this MXF essence container mapping.

8 Carriage of AES3 channel status data and user data

8.1 General

AES3 defines two additional streams of ancillary data, the user data and channel status data. In AES3, these are transmitted one bit per sample per channel for each kind of ancillary data, in the U and C bits respectively, in blocks of 192 samples. Thus, for 2-channel essence, four 192-bit (24-byte) blocks of ancillary are formed. For single channel essence, two blocks are formed.

"User data" as described in this section is specifically the ancillary data recovered from the U bit and does not include any non-audio data encoded in the audio sample bits. Any processing of such encoded data is outside the scope of this spec.

In this standard, ancillary data is not carried within AES3 audio essence elements. Instead it is mapped into the MXF header metadata as described below.

8.2 Block start offset

All blocks of data share a common start point within the AES3 signal, indicated by the Z preamble defined in AES3.

To enable correctly synchronized reinsertion of ancillary data blocks, the sample offset of the first Z preamble in the original input after the start of encoding is stored as an Item in the AES3 audio essence descriptor set.

8.3 Channel status data

8.3.1 Definition

Channel status data bit fields are defined by AES3, which also defines a minimum, standard, and enhanced implementation level.

8.3.2 Channel status data — Minimum implementation

The minimum implementation is defined by AES3-2003 paragraph 7.2.1, and specifies the carriage of bits 0 and 1 of the channel status data only. The remaining bits of the channel status data are indeterminate.

8.3.3 Channel Status Data — Standard Implementation

The standard implementation is defined by AES3-2003 paragraph 7.2.2, and specifies the carriage of bytes 0, 1, 2 and the CRCC of the channel status data.

8.3.4 Channel status data — Enhanced implementation

The enhanced implementation is defined by AES3-2003 paragraph 7.2.3, and specifies the carriage of additional bytes of the channel status data.

SMPTE 337M precludes use of the AES3 enhanced implementation.

Users should be aware that the sample address codes defined by AES3 enhanced implementation for status bytes 14-17 (local) and 18-21 (time-of-day) do not themselves constitute the time code track of an MXF file package. An MXF encoder may optionally interpret the AES data when creating the time code track. An MXF decoder may optionally synthesize AES3 status bytes 14-17 and 18-21 from the time code track.

NOTE – Users should also be aware that time code may be carried in accordance with AES 18 in the user data space.

8.3.5 Mapping of channel status data into MXF

MXF encoders may extract channel status data from the AES3 stream and store it either as a fixed 24 bytes of data within the AES3 audio essence descriptor set or essence described by an additional track in the relevant file package of the header metadata.

NOTE – At some stage in the future the MXF specification may allow metadata streams, such as channel status data, to be mapped directly into the header metadata. At present, variable channel status data may only be represented in the body of the file.

The ChannelStatusDataMode property of the AES3 audio essence descriptor set declares what extraction method the encoder used, for each channel, according to the following table.

Value	Symbol	Meaning
0	NONE	No channel status data is encoded
1	MINIMUM	AES3 Minimum (byte 0 bit 0 = ‘1’)
2	STANDARD	AES3 Standard, derived as below
3	FIXED	Fixed 24 byes of data in FixedChannelStatusData property
4	STREAM	Stream of data within MXF Header Metadata (value reserved for future extension – see Informative Note above)
5	ESSENCE	Stream of data multiplexed within MXF Body

Even though AES3 status bytes 14-17 and 18-21 may be included in an MXF file when ChannelStatusDataMode is STREAM or ESSENCE, these data values should not be interpreted as the time code track of an MXF file package.

The correspondence between properties of the AES3 audio essence descriptor and bits of the AES3 standard channel status implementation (bytes 0,1,2) is as follows:

Byte 0	
Bit 0	const ‘1’; /* professional use */

Bit 1	if (SoundEssenceCoding == kPCM) ‘0’;
	else ‘1’;

Bits 2-4	if (SoundEssenceCoding == kPCM)
	    *value-of(*Emphasis*)*;
	else ‘000’;

Bit 5	if (LockUnlock == TRUE) ‘0’;
	else ‘1’;

Bits 6,7	*value-of(*AES3 Frame Frequency*)*;

Byte 1	

Bits 0-3	if (SoundEssenceCoding == kPCM)
	    *value-of (*ElectroSpatialFormulation*)*
	else ‘0000’;

Bits 4-7	*value-of (*UserDataMode*)*

Byte 2	
Bits 0-2	*value-of (*AuxBitsMode*)*
Bits 3-5	*value-of (*QuantizationBits*)*
Bits 6-7	‘00’ /* alignment level not indicated */

NOTE – The *value-of()* bit field mapping function is defined as the mapping of each bit from the source to the destination in the same relative position, lsb to lsb and so on.

8.4 User data

MXF encoders may extract user data from the AES3 stream and store it either as a fixed 24 bytes of data within the AES3 audio essence descriptor set or as a separate stream of metadata or essence described by an additional track in the relevant file package of the header metadata.

The UserDataMode property of the AES3 audio essence descriptor set declares what extraction method the encoder chose, for each channel. This property is equivalent to AES3 channel status data, byte 1, bits 4-7.

NOTE – At some stage in the future the MXF specification may allow metadata streams, such as user data, to be mapped directly into the header metadata. At present, variable user data may only be represented in the body of the file.

9 SMPTE label for essence container identification

The values for the container UL are given in table 4.

Table 4 – Specification of the essence container label

Byte No.	Description	Value (hex)	Meaning
1-12	Defined by Generic Container		
13	Essence Container Kind	02h	MXF Generic Container
14	Mapping Kind	06h	AES-BWF as listed in SMPTE RP224
15	Content Kind	01h 02h 03h 04h 08h 09h	Wave Frame Wrapped Element Wave Clip Wrapped Element AES Frame Wrapped Element AES Clip Wrapped Element Wave custom Wrapped Element AES custom Wrapped Element as listed in SMPTE RP224
16	Reserved	00h	

Annex A (normative)
Definition of the essence descriptor sets for the essence data defined by this standard

NOTE – The annex uses the following symbols to the left of the table to help identify the entries which link the metadata items together.

- 🗒 Set universal label — top level.
- ↔ Set length — top level.

In all tables describing sets in this annex, the columns are defined as follows:

- Item name: The name of the property.
- Type: The defined type of the property.
- Len: The length of the value in bytes where known.
- Tag: The 2-byte tag of the property when encoded as a KLV local set.
- UL designator: The designator part of the UL key of the property as it is defined in the SMPTE metadata dictionary. In the case of the first row (that defines the set key), the set key is defined in table 2 and the UL designator column is used to specify bytes 14 and 15 of that key.
- Req ?: encoding and decoding requirements as specified in the MXF file format specification.

  NOTE – In some applications, these properties cannot be completed when a file is written. The mechanism for signaling incomplete property values is defined in the MXF file format.

- Meaning: A description of the property.
- Default: A default value which should be used by the decoder if the property is not encoded.
- Example values for ULs are listed in SMPTE RP 224.

A.1 Wave audio essence descriptor

The table below defines the items in the wave audio essence descriptor (a subclass of the generic sound essence descriptor).

	Item Name	**Type**	**Len**	**Local Tag**	**UL Designator**	**Req ?**	**Meaning**	**Default**
🗒	Wave Audio Essence Descriptor	Set UL	16		See table below	Req	Defines the Wave Audio Essence Descriptor Set (a collection of Parametric metadata)	
↔	Length	BER Length	4			Req	Set length	
	All items from the Sound Essence Descriptor in SMPTE377M (File Format Specification Annex D.3) to be included							
	BlockAlign	Uint16	2	3D.0A	04.02.03.02.01	Req	Sample Block alignment	
	SequenceOffset	Uint8	1	3D.0B	04.02.03.02.02	Opt	Zero-based ordinal frame number of first essence data within five-frame sequence (See 5.2)	
	AvgBps	Uint32	4	3D.09	04.02.03.03.05	Req	Average Bytes per second (see 6.2)	
	Channel Assignment	UL	16	3D.32	04.02.01.01.05.00.00.00	Opt	UL enumerating the channel assignment in use eg. SMPTE 320M-A	
	PeakEnvelopeVersion	UInt32	4	3D.29	04.02.03.01.06	Opt	Peak envelope version information (BWF dwVersion)	none
	PeakEnvelopeFormat	UInt32	4	3D.2A	04.02.03.01.07	Opt	Format of a peak point (BWF dwFormat)	none
	PointsPerPeakValue	UInt32	4	3D.2B	04.02.03.01.08	Opt	Number of peak points per peak value (BWF dwPointsPerValue)	none
	PeakEnvelopeBlockSize	UInt32	4	3D.2C	04.02.03.01.09	Opt	Number of audio samples used to generate each peak frame (BWF dwBlockSize)	none
	PeakChannels	UInt32	4	3D.2D	04.02.03.01.0A	Opt	Number of peak channels (BWF dwPeakChannels)	none
	PeakFrames	UInt32	4	3D.2E	04.02.03.01.0B	Opt	Number of peak frames (BWF dwNumPeakFrames)	none

PeakOfPeaksPosition	Position	8	3D.2F	04.02.03.01.0C	Opt	Offset to the first audio sample whose absolute value is the maximum value of the entire audio file (BWF dwPosPeakOfPeaks, extended to 64 bits)	N/A
PeakEnvelopeTimestamp	TimeStamp	8	3D.30	04.02.03.01.0D	Opt	Time stamp of the creation of the peak data (BWF strTimeStamp converted to TimeStamp)	none
PeakEnvelopeData	Stream	N	3D.31	04.02.03.01.0E	Opt	Peak envelope data (BWF peak_envelope_data) Data format is described in 6.4.1	none

The key (UL) for this set is defined below:

Byte No.	Description	Value (hex)	Meaning
1-13	Defined in the Structural Header Metadata Implementation section of SMPTE377M (File Format Specification)		
14	Set Kind (1)	01h	Wave Audio Essence Descriptor
15	Set Kind (2)	48h	
16	Reserved	00h	Reserved

A.2 AES3 audio essence descriptor

The table below defines the items in the AES3 audio essence descriptor (a subclass of the wave audio essence descriptor). This descriptor shall be used when mapping audio essence from an AES3 stream and intentionally preserving the channel status and other parameters of the AES3 stream.

In the table below, the value of N is the value of the ChannelCount property of the AES3 audio essence descriptor. This corresponds to single channel, dual channel, or n channel operation where permitted. The definition of the data properties is given in the normative reference AES3-2003.

Item Name	Type	Len	Local Tag	UL Designator	Req ?	Meaning	Default
AES3 Audio Essence Descriptor	Set UL	16		See table below	Req	Defines the AES3 Audio Essence Descriptor Set (a collection of Parametric metadata)	
Length	BER Length	4			Req	Set length	
All items from the Wave Audio Essence Descriptor in section A.1							
Emphasis	Uint8 (enum)	1	3D.0D	04.02.05.01.06	Opt	AES3 Emphasis (aligned to LSB of this property)	00
BlockStartOffset	Uint16	2	3D.0F	04.02.03.02.03	Opt	AES3 Position of first Z preamble in essence stream (see 8.2)	0
AuxBitsMode	Uint8 (enum)	1	3D.08	04.02.05.01.01	Opt	AES3 Use of Auxiliary Bits (see 8.3)	000
ChannelStatus-Mode	Uint8 (enum)Array	8+N *1	3D.10	04.02.05. 01.02	Opt	AES3 Enumerated mode of carriage of channel status data(see 8.3)	NONE
FixedChannel-StatusData	Array of bytes	8+N *24	3D.11	04.02.05.01.03	Opt	AES3 Fixed data pattern for channel status data(see 8.3)	per AES3 minimum
UserDataMode	Uint8 (enum) Array	8+N *1	3D.12	04.02.05.01.04	Opt	AES3 Enumerated mode of carriage of user data, defined by AES3 section 4. (aligned to LSB of this property) (see 8.3)	0 0
FixedUserData	Array of bytes	8+N *24	3D.13	04.02.05.01.05	Opt	AES3 Fixed data pattern for user data (see 8.3)	0

The key (UL) for this set is defined below:

Byte No.	Description	Value (hex)	Meaning
1-13	Defined in the Structural Header Metadata Implementation section of SMPTE377M (File Format Specification)		
14	Set Kind (1)	01h	AES3 Audio Essence Descriptor
15	Set Kind (2)	47h	
16	Reserved	00h	Reserved

A.3 Wave audio physical descriptor

Item Name	Type	Len	Local Tag	UL Designator	Req ?	Meaning	Default
Wave Audio Physical Descriptor	Set UL	16		See table below	Req	Defines the Wave Audio Physical Descriptor Set (a collection of Parametric metadata copied from the BWF <bext> and <qlty> chunks)	
Length	BER Length	4			Req	Set length	
All items from the Generic Descriptor in SMPTE377M (File Format Specification Table 17) to be included							
CodingHistory	UTF-16 String	N	3D.21	04.02.05.02.01	Opt	Coding History from BWF <bext> chunk (see 6.3)	
FileSecurity Report	UInt32	4	3D.15	04.02.03.02.05	Opt	FileSecurityCode of quality report (see 6.4.2)	
FileSecurity Wave	UInt32	4	3D.16	04.02.03.02.06	Opt	FileSecurityCode of BWF wave data (see 6.4.2)	
BasicData	UTF-16 String	Var	3D.22	04.02.05.02.02.01	Opt	« Basic data » from <qlty> chunk (see 6.4.2)	
StartModulation	UTF-16 String	Var	3D.23	04.02.05.02.03.01	Opt	« Start modulation data » from <qlty> chunk (see 6.4.2)	
QualityEvent	UTF-16 String	Var	3D.24	04.02.05.02.04.01	Opt	« Quality event data » from <qlty> chunk (see 6.4.2)	
EndModulation	UTF-16 String	Var	3D.25	04.02.05.02.05.01	Opt	« End modulation data » from <qlty> chunk (see 6.4.2)	
Quality Parameter	UTF-16 String	Var	3D.26	04.02.05.02.06.01	Opt	« Quality parameter data » from <qlty> chunk (see 6.4.2)	
Operator Comment	UTF-16 String	Var	3D.27	04.02.05.02.07.01	Opt	« Comments of operator » from <qlty> chunk (see 6.4.2)	
CueSheet	UTF-16 String	Var	3D.28	04.02.05.02.08.01	Opt	« Cue sheet data » from <qlty> chunk (see 6.4.2)	
UnknownBWFChunks	Array of Strongref (Unknown Chunk Sets)	8+16*N	3D.33	06.01.01.04.06.0F.00.00	Opt	An array of strong references to Unknown Chunk Sets containing RIFF chunks which were found in the BWF stream, but were unknown to the MXF encoding device at the time of encoding. The Usage is defined in 6.4.4.	

NOTES

1 All properties above are copied from the BWF <bext> chunk or <qlty> chunk or other chunk.

2 For compatibility with all other string properties in MXF, all properties are of type UTF-16 string. The corresponding fields in the BWF <bext> and <qlty> chunk are of type ISO 7 string, which may be further encoded as UTF-8 or shift-JIS in some geographic regions. When the WaveAudioPhysicalDescriptor is used to create BWF files, MXF decoders should restrict the encoded values to those permitted in the particular region.

The key (UL) for this set is defined below:

Byte No.	Description	Value (hex)	Meaning
1-13	Defined in the Structural Header Metadata Implementation section of SMPTE377M (File Format Specification)		
14	Set Kind (1)	01h	Wave Audio Physical Descriptor
15	Set Kind (2)	50h	
16	Reserved	00h	Reserved

A.4 Unknown chunk set

The table below defines the Items in the unknown chunk set. This set is for the carriage of RIFF chunks which were unknown to the MXF encoding device at the time of encoding. Its purpose is to allow transparent, lossless carriage of BWF metadata in MXF systems.

Item Name	Type	Len	Local Tag	UL Designator	Req ?	Meaning	Default
Unknown Chunk	Set UL	16		See table below	Req	Unknown Chunk Set	
Length	BER Length	4			Req	Set length	
Instance UID	UUID	16	3C.0A	01.01.15.02	Req	Unique ID of this instance [**RP210** The ISO/IEC 11578 (Annex A) 16 byte Globally Unique Identifier]	
Generation UID	UUID	16	01.02	05.20.07.01.08	Opt	Generation Identifier [**RP210** Specifies the reference to an overall modification]	
ChunkID	UInt32	4	4F.01	04.06.08.02	Req	The ID of the RIFF Chunk	
ChunkLength	UInt32	4	4F.02	04.06.09.03	Req	The number of bytes of ChunkData	
ChunkData	Stream	var	4F.03	04.07.04	Req	The bytes of the RIFF Chunk data	

The key (UL) for this set is defined below:

Byte No.	Description	Value (hex)	Meaning
1-13	Defined in the Structural Header Metadata Implementation section of SMPTE377M (File Format Specification)		
14	Set Kind (1)	01h	UnknownChunk Set
15	Set Kind (2)	4Fh	
16	Reserved	00h	Reserved

Annex B (informative)
Example of mapping channels and descriptors

Figure B.1 shows a typical OP2a file created by cascading 3 sound clips. In each SouceClip in the material package, the optional channel ID property is used to identify the sound channel to be selected from a multi-channel file package. If the property is absent then all the channels will be used.

The wave audio essence descriptor is used to describe the actual stored essence (e.g., properties such as BlockStartOffset). The wave audio physical descriptor is used to describe properties of the original BWF file such as the CodingHistory property copied from the <bext> chunk.

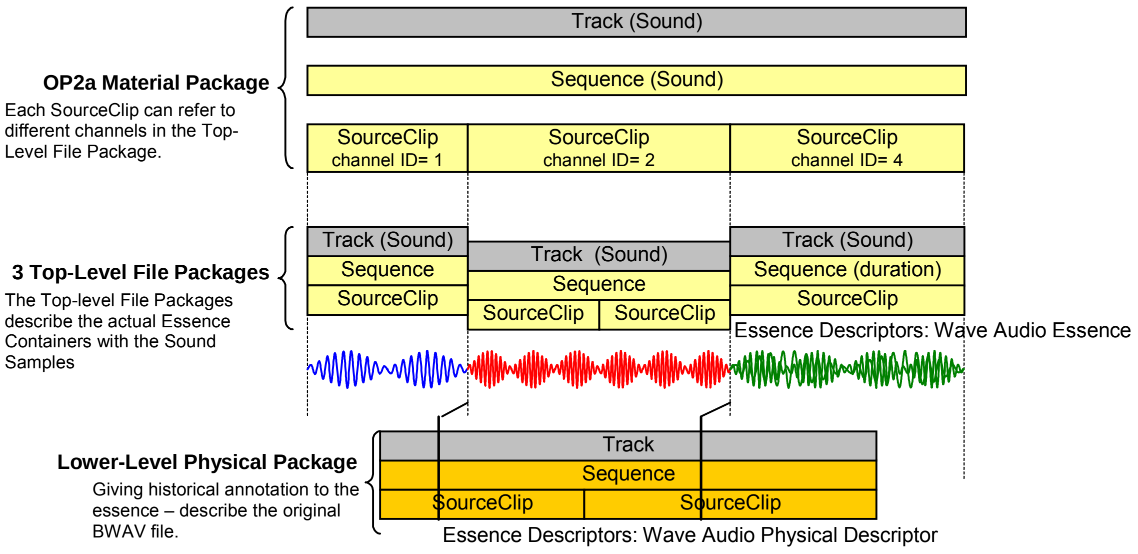
OP2a Material Package
Each SourceClip can refer to different channels in the Top-Level File Package.
Track (Sound)
Sequence (Sound)
SourceClip channel ID= 1
SourceClip channel ID= 2
SourceClip channel ID= 4
3 Top-Level File Packages
The Top-level File Packages describe the actual Essence Containers with the Sound Samples
Track (Sound)
Sequence
SourceClip
Track (Sound)
Sequence
SourceClip
SourceClip
Track (Sound)
Sequence (duration)
SourceClip
Essence Descriptors: Wave Audio Essence
Lower-Level Physical Package
Giving historical annotation to the essence – describe the original BWAV file.
Track
Sequence
SourceClip
SourceClip
Essence Descriptors: Wave Audio Physical Descriptor

Figure B.1 – Package relationships and the use of the physical descriptor

Annex C (informative)
Comparison with other standards

C.1 Comparison with SMPTE 302M and SMPTE 331M

Neither SMPTE 302M nor SMPTE 331M are suitable for efficient mapping of 1 or 2 channels of wave audio essence plus associated metadata.

SMPTE 302M describes packing of 2, 4, 6, or 8 AES3 channels into an MPEG transport stream along with packing of V, U, and C bits. The sample packing, whilst efficient for MPEG systems, is not at all straightforward for byte-oriented processors. There are no provisions for carriage of clip-oriented metadata.

SMPTE 331M describes audio elements which contain eight AES3 channels. Each sample is stored in 4 bytes and includes V, U, and C bits. Unused channels are flagged, but the size of the element remains constant no matter how many channels are in use. SMPTE 331M itself does not carry clip-oriented metadata, although this is described fully in companion documents.

C.2 Comparison with SMPTE 337M / SMPTE 338M / SMPTE339M and related standards

SMPTE 337M defines the mapping of non-PCM data of a number of types into an AES3 stream, and the multiplexing of these types. This standard defines the carriage of SMPTE 337M data in an MXF essence container, without multiplexing.

SMPTE 338M defines a number of types of non-PCM data. This standard defines the mapping of SMPTE 338M data type definitions into the sound essence coding property. Note that some SMPTE 338M data types identify proprietary compressed audio data streams.

SMPTE 339M defines the format of a SMPTE 12M and SMPTE 309M time code packet within SMPTE 337M data _stream_number 7. This standard defines the carriage of SMPTE 339M time stamp data in the MXF time code track.

Annex D (informative)
Bibliography

ANSI/SMPTE 298M-1997, Television — Universal Labels for Unique Identification of Digital Data

SMPTE 12M-1999, Television, Audio and Film — Time and Control Code

SMPTE 302M-2002, Television — Mapping of AES3 data into an MPEG Transport Stream

SMPTE 309M-1999, Television — Transmission of Date and Time Zone Information in Binary Groups of Time and Control Code

SMPTE 314M-1999, Television — Data Structure for DV-Based Audio, Data and Compressed Video Television — 25 and 50 Mb/s

SMPTE 320M-1999, Television — Channel Assignments and Levels on Multichannel Audio Media

SMPTE 323M-2004, Motion-Picture Film — Channel Assignments and Levels on Multichannel Audio Media

SMPTE 330M-2000, Television — Unique Material Identifier (UMID)

SMPTE 331M-2000, Television — Element and Metadata Definitions for SDTI-CP

SMPTE 336M-2000, Television — Data Coding Protocol using Key-Length-Value

SMPTE 380M, Television — Material Exchange Format (MXF) — Descriptive Metadata Scheme 1

SMPTE RP 205-2000, Application of Unique Material Identifiers in Production and Broadcast Environments

SMPTE RP 210, Metadata Dictionary Registry of Metadata Element Descriptions

SMPTE EG 41-2004, Television — Material Exchange Format (MXF) — Engineering Guideline

AES18-1996 (R2002), Digital Audio Engineering — Format for the User Data Channel of the AES Digital Audio Interface

EBU Recommendation R98-1999, Format for CodingHistory Field in Broadcast Wave Format Files, BWF

EBU Recommendation R99-1999, Unique Source Identifier (USID) for Use in the OriginatorReference Field of the Broadcast Wave Format

EBU Tech 3285 Supplement 1-2001, Specification of the Broadcast Wave Format — A Format for Audio Data Files in Broadcasting — MPEG Audio

EBU/SMPTE Task Force for Harmonized Standards for the Exchange of Programme Material as Bitstreams, Final Report: Analyses and Results, Sept 1998

Japanese Post Production Association, "Broadcast Wave Format (Japan), (BWF-J)"