SMPTE ST 382:2023
Revision of SMPTE ST 382:2007

SMPTE STANDARD

Material Exchange Format — Mapping AES3 and Broadcast Wave Audio into the MXF Generic Container

Table of contents

Approved June 8, 2022

Foreword

SMPTE (the Society of Motion Picture and Television Engineers) is an internationally-recognized standards developing organization. Headquartered and incorporated in the United States of America, SMPTE has members in over 80 countries on six continents. SMPTE's Engineering Documents, including Standards, Recommended Practices, and Engineering Guidelines, are prepared by SMPTE's Technology Committees. Participation in these Committees is open to all with a bona fide interest in their work. SMPTE cooperates closely with other standards-developing organizations, including ISO, IEC and ITU.

SMPTE Engineering Documents are drafted in accordance with the rules given in its Standards Operations Manual. This SMPTE Engineering Document was prepared by Technology Committee 31FS.

This revision consolidates amendments and revises document references.

1 Scope

This standard defines the mapping of digital audio data, ancillary data and metadata from the broadcast wave format (BWF) and from 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 can be uncompressed PCM audio data, compressed data or raw data as in BWF, AES3 or SMPTE ST 337 carried in a single AES3 stream. 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 the Labels Register at SMPTE ST 2123.

In order to achieve interoperability within any given operational pattern, restrictions can 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 Conformance Notation

Normative text is text that describes elements of the design that are indispensable or contains the conformance language keywords: "shall", "should", or "may". Informative text is text that is potentially helpful to the user, but not indispensable, and can be removed, changed, or added editorially without affecting interoperability. Informative text does not contain any conformance keywords.

All text in this document is, by default, normative, except: the Introduction, any clause explicitly labeled as "Informative" or individual paragraphs that start with "Note:"

The keywords "shall" and "shall not" indicate requirements strictly to be followed in order to conform to the document and from which no deviation is permitted.

The keywords, "should" and "should not" indicate that, among several possibilities, one is recommended as particularly suitable, without mentioning or excluding others; or that a certain course of action is preferred but not necessarily required; or that (in the negative form) a certain possibility or course of action is deprecated but not prohibited.

The keywords "may" and "need not" indicate courses of action permissible within the limits of the document.

The keyword "reserved" indicates a provision that is not defined at this time, shall not be used, and may be defined in the future. The keyword "forbidden" indicates "reserved" and in addition indicates that the provision will never be defined in the future.

Unless otherwise specified the order of precedence of the types of normative information in this document shall be as follows. Normative prose shall be the authoritative definition. Tables shall be next, followed by formal languages, then figures, and then any other language forms.

3 Normative References

The following documents contain provisions that, through reference in this text, constitute provisions of this standard. Dated references require that the specific edition cited shall be used as the reference. Undated citations refer to the edition of the referenced document (including any amendments) current at the date of publication of this document. All documents are subject to revision, and users of this engineering document are encouraged to investigate the possibility of applying the most recent edition of any undated reference.

SMPTE ST 337, Format for Non-PCM Audio and Data in an AES3 Serial Digital Audio Interface

SMPTE ST 338, Format for Non-PCM Audio and Data in AES3 — Data Types

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

SMPTE ST 377-1, Material Exchange Format (MXF) — File Format Specification

SMPTE ST 379 (all parts), Material Exchange Format (MXF) — MXF Generic Container

SMPTE ST 2123, SMPTE Metadata Registers

AES3-2009, AES standard for digital audio engineering — Serial Transmission Format for Two-Channel Linearly Represented Digital Audio Data

AES31-2, AES Standard on Network and File Transfer of Audio — Audio-File Transfer and Exchange — File Format for Transferring Digital Audio Data Between Systems of Different Type and Manufacture

4 Terms, definitions, symbols and abbreviated terms

For the purposes of this document, the terms, definitions, symbols and abbreviated terms given in SMPTE ST 377-1, SMPTE ST 379 (all parts), AES3-2009 and the following apply:

4.1
AES

Audio Engineering Society

4.2
BEXT

Broadcast Extension Chunk

4.3
block

group of 192 consecutive frames.

Note 1 to entry: The start of a block is designated by a special subframe preamble.

4.4
BWF

Broadcast Wave Format

4.5
BWFF

Broadcast Wave Format file as defined in AES31-2

4.6
CRCC

Cyclic Redundancy Check Code

4.7
frame

sequence of two successive and associated subframes as described in 4.1.2 of AES3-2009

4.8
LEVL

Peak Envelope (level) RIFF Chunk defined by Annex G

4.9
PCM

Pulse Code Modulation

4.10
Resource Interchange File Format
RIFF

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

4.11
subframe

fixed structure used to carry the information described in 4.1.1 and 4.1.2 of AES3-2009

4.12
QLTY

capturing report (quality) RIFF Chunk defined by Annex H

4.13
char[ ]

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.

Note 1 to entry: In some geographic areas, strings of type char[ ] can contain UTF-8 or Shift-JIS characters.

5 Introduction

This clause is entirely informative and does not form an integral part of this Engineering Document.

This standard defines the mapping of digital audio data and ancillary data and metadata from broadcast wave format (BWF) files, and 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 files. 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 Clause 6.

This standard defines wave audio essence elements which can 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 6.2.

Data carried in wave audio essence elements can be uncompressed PCM audio data or compressed data, as defined in AES31-2, AES3-2009 or SMPTE ST 337 carried in a single AES3 stream. The specific format is indicated by the Sound Essence Coding property, which is described in 7.2.1 of this document.

BWF defines metadata which applies to the entire clip. This standard maps this metadata into the MXF header metadata. This is described in Clause 7.

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 Clause 9.

NOTE Data can be lost if applications do not support the carriage of the broadcast wave metadata or AES metadata parameters that are not mapped to the MXF structural metadata. It is recommended that implementers and users 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 BWFF chunk parameters as described in 7.3.

6 Packaging 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. Source Clips in the material package may reference a given channel or channels within the top-level file package by setting an optional Channel ID 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. Clause 6.1 considers the validity bit, while Clause 9 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 7.2.

NOTE AES31-2 provides additional details of the mapping of audio samples into data bytes. SMPTE ST 2035 and SMPTE ST 323 provide definitions of commonly used assignments of multiple channels. Rec. ITU-R BS.1352 also has details of the mapping of audio samples into data bytes.

MXF encoders should set the optional Channel Assignment property of the Wave Audio Essence Descriptor (see Table A.1) to the appropriate Label from the SMPTE Labels Register (i.e., the register defined by SMPTE ST 2123) in order to describe specific channel assignments.

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

6.1 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. When the validity bit is not present, applications shall process audio data as though the validity bit were set to zero.

This data is not available in the packed wave audio essence elements data format, and all applications shall process audio data as though the validity bit were set to zero.

NOTE SMPTE ST 314:2015, paragraph 4.6.2.1.3 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. It is recommended that these samples be overwritten with valid data in the data stream before packing into wave audio essence elements. This applies only to DV-derived audio; for AES3-derived audio, 8000h is valid sample data. DV-derived audio sample data can be determined from the Sound Essence Coding parameter which can be extended using the Labels Register (defined by SMPTE ST 2123) to include DV-derived audio types.

6.2 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.

If the audio is locked to video and if the audio sample rate (i.e., the sampling rate for linear audio or the rate of audio access units for compressed audio) is an integer multiple of the video frame rate, then the audio element shall have the same size in each content package.

EXAMPLE 1
In a multiplexed, frame-wrapped essence container that carries video with a frame rate of 25 frames per second, and locked linear audio with a sample rate of 48 kHz, each audio element contains exactly 1920 samples.

If the audio sample rate is not an integer multiple of the video frame rate, then the number of samples in each element shall vary such that a correct aggregate rate is maintained.

EXAMPLE 2
In a multiplexed, frame-wrapped essence container that carries video with a frame rate of 30*1000/1001 (≈ 29.97) frames per second, and locked linear audio with a sample rate of 48 kHz, the aggregate ratio of audio samples per video frame is 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. However, other sequences are also possible.

The ordinal number of the first frame of audio essence within a five-frame sequence may optionally be indicated in the Wave Audio Essence Descriptor :: Sequence Offset property (see A.1), to provide guidance in processing.

NOTE 1 In the case where the audio sample rate is not an integer multiple of the video frame rate, then there will be a variable number of samples per video frame which could require the audio samples to be indexed as variable rate elements. This can be avoided by adding fill KLV items to ensure constant bytes per element.

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 ST 379 (all parts).

NOTE 2 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 Block Align property of the Wave Audio Essence Descriptor.

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

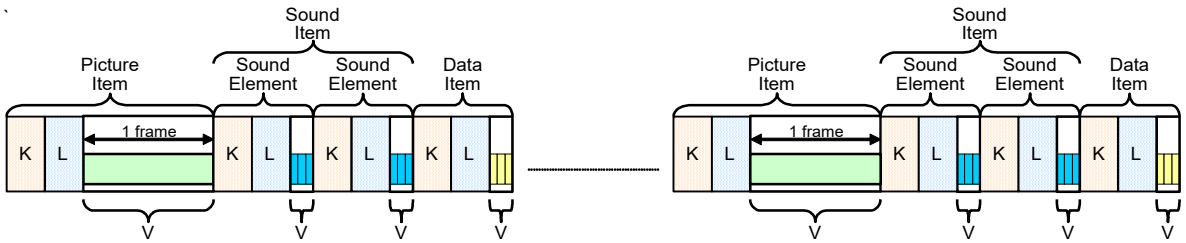

Figure 1 — Representation of frame wrapping with other elements present.

The generic container key for the sound item is given in 6.5.1.

6.3 Division of Audio Essence into Clip-Wrapped Content Packages

An alternative to frame wrapping is to encode all of the audio data in a single element. This is defined as clip-wrapped mapping in SMPTE ST 379 (all parts).

This wrapping may also be useful in sound-only applications, as shown in Figure 2.

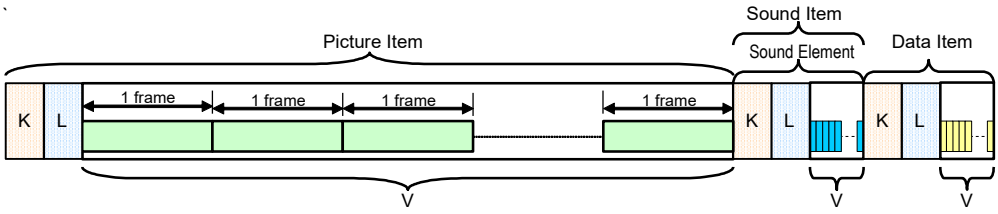

Figure 2 — Simple representation of clip wrapping with other elements.

6.4 Custom-Wrapped Audio Elements

When audio elements are created that are neither frame wrapped nor clip wrapped, the file shall use KLV Key values that indicate custom wrapping.

When the audio wrapping isbound to some external criteria where the “approximately constant” number of samples per element cannot be guaranteed, the file shall use key values that indicate custom wrapping.

EXAMPLE

The case where a logical pattern of video frames occurs: When interleaving audio with MPEG-2 video, the size of the GOP (group of pictures) need not be constant, and it could be desirable for the arrangement of audio elements within a content package to correspond to each GOP. For example, Figure 3 shows the case where the first GOP has a single frame, while the second GOP has only three frames. The length of audio elements could be chosen in lengths approximately equal to these logical frame groups.

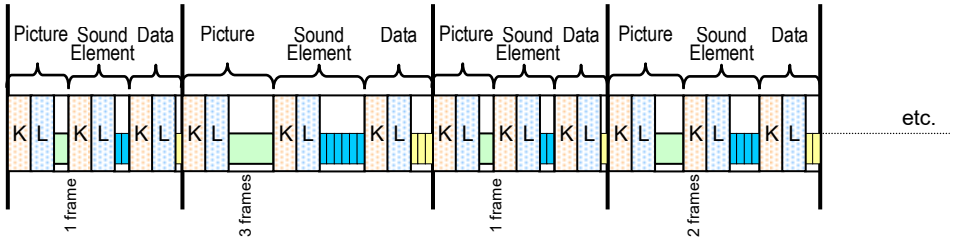

Figure 3 — Simple representation of custom wrapping with other elements.

Constant duration audio Custom-Wrapping specifies that all audio elements shall contain exactly the same number of audio samples, such as one or two seconds or constant number of video frames. The number of audio samples may be different in the first and last elements in a file.

The constant duration custom-wrapped audio element shall use key values that indicate Custom-Wrapping and the essence shall be signaled by the Essence Container Labels for the constant duration Custom-Wrapped elements listed in Table 6.

6.5 KLV Coding of Digital Audio Data Sound Elements

6.5.1 Wave sound element key

The values of the first 12 bytes of the essence element key are defined in SMPTE ST 379 (all parts), 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		The number (used as an index) of this Sound Element in this Generic Container

6.5.2 Essence element count — Byte 14

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

6.5.3 Essence element type — Byte 15

The values are given in Table 1.

6.5.4 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 ST 379 (all parts), which shall remain constant throughout any instance of a generic container.

6.5.5 Wave sound element length

For frame wrapping, the length field of the KLV-coded element shall be 4 bytes BER long-form encoded (i.e., 83h.xx.yy.zz).

For clip 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 custom wrapping, the length field of the KLV-coded element shall be any value within the range defined by SMPTE ST 377-1.

NOTE SMPTE ST 377-1 requires MXF decoders to support BER long-form encodings with other byte lengths.

6.5.6 Wave sound element value

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

In a frame-wrapped file containing uncompressed audio samples, the start position of the first sample of wave audio data shall 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 or because of audio encoding schemes that encode multiple audio samples in non-separable blocks of bits (i.e., audio frames), the start position of the video access unit should fall within the duration of the first sample or audio frame of the sound element.

6.6 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 and other SMPTE specifications applicable to the Essence Container and Operational Pattern.

6.7 Optional Mapping of Material Package Tracks to Audio Channel IDs

Annex E specifies the semantics of Channel ID (per SMPTE ST 377-1) for audio essence conforming to this specification, thereby providing a simple mechanism for mapping one or two channels of audio from a Source Package. It provides an optional mechanism for an MXF decoder to “play” a subset of stored audio channels. A complete N to M audio channel mapping mechanism is outside the scope of this specification.

7 Carriage of BWF Metadata Chunks

Clause 7 applies only to implementations where BWF metadata is available.

7.1 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 AES31-2). Defined chunks include the <fmt> chunk and the broadcast wave extension <bext> chunk. Other RIFF chunks are allowed.

A minimum implementation shall map all of the RIFF chunks that are required by the AES31-2 specification.

A transparent implementation shall map all of the RIFF chunks in a BWF file.

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.

7.2 <fmt> Chunk

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

```
typedef struct {
   UInt16 wFormatTag;
   Uint16 nChannels;
   Uint32 nSamplesPerSec;
   Uint32 nAvgBytesPerSec;
   Uint16 nBlockAlign;
   Uint16 nBitsPerSample;
} fmt-ck;
```

The precise definitions of these parameters and other coding rules are given in AES31-2-2006 Annex A, and readers should refer to that standard. These are also listed in Annex D in this document.

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 Average Bytes Per Second.

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:

$$\mathrm{nBlockAlign} = \mathrm{nChannels} \cdot \mathrm{floor}\left(\frac{\mathrm{nBitsPerSample} + 7}{8}\right)$$

and the reverse is true:

$$\mathrm{nChannels} = \frac{\mathrm{nBlockAlign}}{\mathrm{floor}\left(\frac{\mathrm{nBitsPerSample} + 7}{8}\right)}$$

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

For SMPTE ST 337, the number SubframeCount of AES channels (aka AES subframes) present in the essence data is:

$$\mathrm{SubframeCount} = \frac{\mathrm{nBlockAlign}}{\mathrm{floor}\left(\frac{\mathrm{nBitsPerSample} + 7}{8}\right)}$$

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

nBitsPerSample 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 when mapping it to an MXF property.

For MXF, the parameters in Table 2 are carried as individual items in the Wave Audio Essence Descriptor set. The correspondence is as follows:

Table 2 — <fmt> Chunk to MXF mapping.

Field in <fmt> chunk	MXF Set :: Property
wFormatTag	Wave Audio Essence Descriptor :: Sound Essence Coding
nChannels	Wave Audio Essence Descriptor :: Channel Count
nSamplesPerSec	Wave Audio Essence Descriptor :: Audio Sampling Rate
nAvgBytesPerSec	Wave Audio Essence Descriptor :: Average Bytes Per Second
nBlockAlign	Wave Audio Essence Descriptor :: Block Align
nBitsPerSample	Wave Audio Essence Descriptor :: Quantization Bits

The Wave Audio Essence Descriptor :: Sample Rate property is defined by SMPTE ST 377-1 in the File Descriptor. Its value is most commonly set to the frame rate of the video when in an interleave of video and audio. When in an audio-only file, it should be set to the same value as Wave Audio Essence Descriptor :: Audio Sampling Rate, which is defined by SMPTE ST 377-1 in the generic Sound Essence Descriptor.

Any <format-specific-fields> that are present in a BWF file but are not specified in AES31-2 are not carried in MXF.

7.2.1 Sound Essence Compression

The Sound Essence Coding property of the Wave Audio Essence Descriptor in MXF is a 16-byte SMPTE UL. The equivalent field in the <fmt> chunk is wFormatTag.

When the essence container element contains uncompressed (PCM) audio that employs the same bit-by-bit packing of sample data as is used by AES31-2, this optional property should be absent from the descriptor set.

For all other sound essence compressions (e.g., MPEG 1 level 1, DV or other coding formats specified in AES31-2, SMPTE ST 338) an appropriate value from the Labels Register (defined by SMPTE ST 2123) shall be used.

If the Sound Essence Coding property is not present in the Descriptor, MXF decoders shall use the Item Designator value of 04.02.02.01.01.00.00.00.

NOTE This corresponds to the case of uncompressed audio essence that employs the same bit-by-bit packing of sample data as is used by AES31-2.

7.3 Broadcast Extension <bext> Chunk

The <bext> chunk defined by AES31-2 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];
    char Reserved[190];
    char CodingHistory[ ];
} bext-ck;
```

The precise definitions of these parameters and other coding rules are given in AES31-2, and readers should refer to that standard. These definitions are also listed in Annex D of this document.

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 R 099.
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 AES31-2.
Version	is the version number of the BWF format, currently defined as 0001h.
UMID	is the material UMID per SMPTE ST 330. 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 AES31-2.

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.

AES31-2 does not define any extensions to the <bext> chunk. Any <bext> chunk extensions present in a BWF file are not carried in MXF.

NOTE The first 32 bytes of the UMID (i.e., the basic UMID) are mapped into the MXF source package's Package UID property to identify the essence. The remaining 32 bytes of the extended UMID can 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**	Clip Framework :: Annotation :: Annotation Description
opt	**Originator**	Clip Framework :: Contacts List :: Person or Clip Framework :: Contacts List :: Organization
opt	**OriginatorReference**	Clip Framework :: Clip Number
req	**OriginationDate**	File Package :: Package Creation Date
req	**OriginationTime**	File Package :: Package Creation Date
req	**TimeReference(Low/High)**	File Package :: Audio Track:: Origin NOTE The value stored is the negative value of TimeReference so that the sign of the Origin parameter is correct according to SMPTE ST 377-1.
	Version	(unused)
req	**UMID[0 – 31]**	Source Package :: Package UID This UMID shall be used in the package containing the Wave Audio Physical Descriptor.
	Reserved	(unused)
req	**CodingHistory**	Physical Package :: Wave Audio Physical Descriptor :: Bext Coding History

7.4 Other Chunks

Other optional RIFF chunks may be present in a BWF file.

7.4.1 Peak envelope <levl> chunk

The <levl> chunk as defined in Annex G contains a peak envelope for the audio essence. The <levl> chunk is carried as optional properties of the Wave Audio Essence Descriptor set (see A.1).

MXF encoders shall copy all parameters of the <levl> chunk to the associated properties of the Wave Audio Essence Descriptor set. When recreating a BWF file, MXF decoders shall reconstruct the <levl> chunk, or shall retransmit the original chunk.

The peak_envelope_data in the <levl> chunk is stored little-endian in the Peak Envelope Data property of the Wave Audio Essence Descriptor. Since this chunk contains an essence-like waveform, the data in this chunk shall not be reordered to the big-endian format of MXF metadata. The contents of this chunk shall be treated as a string of bytes.

7.4.2 Quality <qlty> chunk

The optional <qlty> chunk is defined in Annex H and is carried as properties of the optional Wave Audio Physical Descriptor.

When receiving a BWF file, MXF encoders shall preserve the coding history field of the <bext> chunk and all of the fields of the <qlty> chunk as individual properties in the Wave Audio Physical Descriptor in a lower-level physical package.

Encoders may create properties in the descriptive metadata from the <bext> and <qlty> chunks according to the translations in the table in 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. A Wave Audio Physical Descriptor may still be used for dark chunk carriage.

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

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 Wave Audio Physical Descriptor, if present.

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

7.4.3 Other chunks

For MXF, these chunks are treated as dark metadata and are carried as optional properties of the Wave Audio Physical Descriptor 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 Wave Audio Physical Descriptor, a data element would be registered in SMPTE ST 2123 and would have a SMPTE UL corresponding to the property dwHeadBitrate having the data type DWORD. The primer pack mechanism in SMPTE ST 377-1 would be used to allocate this property a 2-byte dynamic tag which would then be used to insert the data value into the Wave Audio Physical Descriptor. An encoder or decoder wishing to use this property must know its SMPTE 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.

7.4.4 Unknown chunks while MXF encoding

An MXF encoder should carry unknown BWF chunks in Unknown Chunk sets (Clause 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 Wave Audio Physical Descriptor property "Unknown BWF Chunks" 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.

7.4.5 Use of a physical descriptor (Informative)

Why is a physical descriptor used to hold the AES / BWFF source properties and not a file descriptor? This is because the source reference chain of the MXF data model (as defined in SMPTE ST 377-1) is used to contain metadata which gives an audit trail of what happened to the file in the AAF-MXF environment. This linkage between source packages is essentially an MXF centric view of the world. When content enters into the MXF environment from a non-MXF source, e.g., a physical tape, a physical file on disc such as BWF, AVI or other external formats, a physical descriptor 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 file descriptor 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 physical descriptor.

8 Carriage of SMPTE ST 339 Time Stamp, V-Sync and User Data

Clause 8 applies only to implementations where SMPTE ST 339 time stamps, V-sync bursts or user data are available.

SMPTE ST 339 defines a time stamp data burst, including SMPTE ST 12-1 time code and SMPTE ST 309 time and date information, plus timing offset information.

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

In the MXF header metadata, time stamp data should be carried in the time code sets of a time code track of the MXF file package. An MXF file package should include a time code track for the time stamp data of each time-stamped SMPTE ST 337 data stream.

There is a possibility that the time stamp data could result in discontinuous time code. An MXF encoder should represent any discontinuities by concatenating Timecode Segments on an MXF Timecode Track.

Provisions for multiple time stamp data streams are given in Annex F.

The additional timing precision provided by word 4 sample_count and word 8 reference_position of the SMPTE ST 339 time stamp data burst shall be carried in the Track::Origin 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 converted into the units of the Track::Origin property in accordance with SMPTE ST 377-1.

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

In an MXF essence container, V-sync bursts shall be taken as the reference point for the start of edit units. These reference points shall be used to achieve the desired wrapping and indexing strategy for the MXF file (e.g., clip wrapping or frame wrapping). V-sync bursts shall not be deleted from the essence data.

The spacing of V-sync bursts within the essence stream may be used to derive the Sequence Offset property of the Wave Audio Essence Descriptor.

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

9 Carriage of AES3 Channel Status Data and User Data

Clause 9 applies only to implementations where AES3 channel status data or user data is available.

9.1 General

AES3-2009 defines two additional streams of ancillary data, the user data and channel status data. In AES3-2009, 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 standard.

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.

9.2 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 shall be stored in the AES3 Audio Essence Descriptor :: Block Start Offset.

9.3 Channel Status Data

9.3.1 Definition

Channel status data bit fields are defined by AES3-2009, which also defines a minimum, standard, and enhanced implementation level.

9.3.2 Channel status data — Minimum implementation

The minimum implementation is defined by AES3-2009, 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 not specified.

9.3.3 Channel status data — Standard implementation

The standard implementation is defined by AES3-2009, paragraph 7.2.2, and specifies the carriage of bytes 0, 1, 2 and the CRCC of the channel status data.

9.3.4 Channel status data — Enhanced implementation

The enhanced implementation is defined by AES3-2009, paragraph 7.2.3, and specifies the carriage of additional bytes of the channel status data.

NOTE SMPTE ST 337 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 Timecode Track of an MXF file package. An MXF encoder may optionally interpret the AES data when creating the Timecode Track. An MXF decoder may synthesize AES3 status bytes 14-17 and 18-21 from the Timecode Track.

9.3.5 Optional 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 Variable channel status data can only be represented in the body of the file.

The enumerated Channel Status Mode property of the AES3 Audio Essence Descriptor set declares what extraction method the encoder used, for each channel, according to the following, in Table 4:

Table 4 — Enumerated values of Channel Status Mode.

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 bytes of data in Fixed Channel Status Data property
4	STREAM	SMPTE reserved
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 Channel Status Mode is STREAM or ESSENCE, these data values should not be interpreted as the Timecode 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 in Table 5:

Table 5 — Mapping of AES3 Audio Essence Descriptor properties to channel status data.

Byte 0	
Bit 0	const '1'; /* professional use */
Bit 1	if (Sound Essence Coding == kPCM) '0'; else '1';
Bits 2-4	if (Sound Essence Coding == kPCM) *value-of(*Emphasis*)*; else '000';
Bit 5	if (Locked == TRUE) '0'; else '1';
Bits 6,7	*value-of(*AES3 Frame Frequency*)*;

Byte 1	
Bits 0-3	if (Sound Essence Coding == kPCM) *value-of (*Electrospatial Formulation*)* *else '0000';*
Bits 4-7	*value-of (*User Data Mode*)*

Byte 2	
Bits 0-2	*value-of (*Auxiliary Bits Mode*)*
Bits 3-5	*value-of (*Quantization Bits*)*
Bits 6-7	'00' /* alignment level not indicated */

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.

9.4 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 AES3 Audio Essence Descriptor :: User Data Mode property shall specify the extraction method the encoder chooses, for each channel. This property shall be set to the AES3 channel status data, byte 1, bits 4-7.

NOTE Variable user data can only be represented in the body of the file.

10 SMPTE Label for Essence Container Identification

The values for the container UL are given in Table 6.

Table 6 — 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
15	Content Kind	01h 02h 03h 04h 08h 09h 0Ah 0Bh	Wave Frame-Wrapped Element Wave Clip-Wrapped Element AES Frame-Wrapped Element AES Clip-Wrapped Element Wave Custom-Wrapped Element AES Custom-Wrapped Element Wave constant duration Custom-Wrapped Element AES constant duration Custom-Wrapped Element
16	Reserved	00h	

Annex A (Normative)
Definition of the Essence Descriptor Sets for the Essence Data Defined by this Standard

Annex A uses the following symbols to the left of the Table A.1, Table A.3, Table A.5, and Table A.7 to help identify the entries that link the metadata items together.

Set universal label — top level.

↔ Set length — top level.

In all tables describing sets in this annex (Table A.1, Table A.3, Table A.5, and Table A.7), 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 the specified table and the UL designator column is used to specify bytes 14 and 15 of that key.
- **Req ?:** Encoding and decoding requirements as specified in SMPTE ST 377-1.
- **Meaning:** A description of the property.
- **Default:** A default value which should be used by the decoder if the property is not encoded.

A.1 Wave Audio Essence Descriptor

Table A.1 defines the items in the Wave Audio Essence Descriptor (a subclass of the generic sound essence descriptor).

Table A.1 — Wave Audio Essence Descriptor.

	Item Name	Type	Len	Local Tag	UL Designator	Req ?	Meaning	Default
	Wave Audio Essence Descriptor	Set UL	16		See Table A.2	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 SMPTE ST 377-1 to be included							
	Block Align	Uint16	2	3D.0A	04.02.03.02.01	Req	Sample Block alignment	
	Sequence Offset	Uint8	1	3D.0B	04.02.03.02.02	Opt	Zero-based ordinal frame number of first essence data within five-frame sequence (see 6.2)	
	Average Bytes Per Second	Uint32	4	3D.09	04.02.03.03.05	Req	Average Bytes per second (see 6.2)	

Item Name	Type	Len	Local Tag	UL Designator	Req ?	Meaning	Default
Channel Assignment	UL	16	3D.32	04.02.01.01.05.00.00.00	Opt	UL enumerating the channel assignment in use	
Peak Envelope Version	UInt32	4	3D.29	04.02.03.01.06	Opt	Peak envelope version information (dwVersion of the <levl> chunk)	none
Peak Envelope Format	UInt32	4	3D.2A	04.02.03.01.07	Opt	Format of a peak point (dwFormat of the <levl> chunk)	none
Points Per Peak Value	UInt32	4	3D.2B	04.02.03.01.08	Opt	Number of peak points per peak value (dwPointsPerValue of the <levl> chunk)	none
Peak Envelope Block Size	UInt32	4	3D.2C	04.02.03.01.09	Opt	Number of audio samples used to generate each peak frame (dwBlockSize of the <levl> chunk)	none
Peak Channels	UInt32	4	3D.2D	04.02.03.01.0A	Opt	Number of peak channels (dwPeakChannels of the <levl> chunk)	none
Peak Frames	UInt32	4	3D.2E	04.02.03.01.0B	Opt	Number of peak frames (dwNumPeakFrames of the <levl> chunk)	none
Peak Of Peaks Position	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 (dwPosPeakOfPeaks of the <levl> chunk, but extended to 64 bits)	N/A
Peak Envelope Timestamp	TimeStamp	8	3D.30	04.02.03.01.0D	Opt	Time stamp of the creation of the peak data (strTimeStamp of the <levl> chunk, but converted to TimeStamp)	none
Peak Envelope Data	Stream	N	3D.31	04.02.03.01.0E	Opt	Peak envelope data (peak_envelope_data of the <levl> chunk) Data format is described in 7.4.1.	none

The key (UL) for this set is defined in Table A.2.

Table A.2 — Key for Wave Audio Essence Descriptor.

Byte No.	Description	Value (hex)	Meaning
1-13	Defined in the Structural Header Metadata Implementation clause of SMPTE ST 377-1		
14	Set Kind (1)	01h	Wave Audio Essence Descriptor
15	Set Kind (2)	48h	
16	Reserved	00h	Reserved

A.2 AES3 Audio Essence Descriptor

Table A.3 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 Table A.3, the value of N is the value of the Channel Count 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.

Table A.3 — AES3 Audio Essence Descriptor.

Item Name	Type	Len	Local Tag	UL Designator	Req ?	Meaning	Default
AES3 Audio Essence Descriptor	Set UL	16		See Table A.4	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 A.1							
Emphasis	Uint8 (enum)	1	3D.0D	04.02.05.01.06	Opt	AES3 Emphasis (aligned to LSB of this property)	00
Block Start Offset	Uint16	2	3D.0F	04.02.03.02.03	Opt	AES3 Position of first Z preamble in essence stream (see 9.2)	0
Auxiliary Bits Mode	Uint8 (enum)	1	3D.08	04.02.05.01.01	Opt	AES3 Use of Auxiliary Bits (see 9.3)	000
Channel Status 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 9.3)	NONE
Fixed Channel Status Data	Array of bytes	8+N*24	3D.11	04.02.05.01.03	Opt	AES3 Fixed data pattern for channel status data (see 9.3)	per AES3 minimum
User Data Mode	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 . (aligned to LSB of this property) (see 9.3)	0 0
Fixed User Data	Array of bytes	8+N*24	3D.13	04.02.05.01.05	Opt	AES3 Fixed data pattern for user data (see 9.3)	0
Linked Timecode Track ID	Uint32	4	dyn	04.02.05.01.07	Opt	AES3 association of SMPTE ST 337 time stamp to Timecode Track ID	Omitted
SMPTE ST 337 Data Stream Number	Uint8	1	dyn	04.02.05.01.08	Opt	The data_stream_number of the SMPTE ST 337 data stream being described (if applicable)	Omitted

The key (UL) for this set is defined in Table A.4.

Table A.4 — Key for AES3 Audio Essence Descriptor.

Byte No.	Description	Value (hex)	Meaning
1-13	Defined in the Structural Header Metadata Implementation clause of SMPTE ST 377-1		
14	Set Kind (1)	01h	AES3 Audio Essence Descriptor
15	Set Kind (2)	47h	
16	Reserved	00h	Reserved

A.3 Wave Audio Physical Descriptor

Table A.5 — Wave Audio Physical Descriptor.

Item Name	Type	Len	Local Tag	UL Designator	Req ?	Meaning	Default
Wave Audio Physical Descriptor	Set UL	16		See Table A.6	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 SMPTE ST 377-1 to be included							
Bext Coding History	UTF-16 String	N	3D.21	04.02.05.02.01.01	Opt	Coding History from BWF <bext> chunk (see 7.3)	
Qlty File Security Report	UInt32	4	3D.15	04.02.03.02.05	Opt	FileSecurityCode (checksum) of quality report (see 7.4.2)	
Qlty File Security Wave	UInt32	4	3D.16	04.02.03.02.06	Opt	FileSecurityCode (checksum) of BWF wave data (see 7.4.2)	
Qlty Basic Data	UTF-16 String	Var	3D.22	04.02.05.02.02.01	Opt	« Basic data » from <qlty> chunk (see 7.4.2)	
Qlty Start Of Modulation	UTF-16 String	Var	3D.23	04.02.05.02.03.01	Opt	« Start modulation data » from <qlty> chunk (see 7.4.2)	
Qlty Quality Event	UTF-16 String	Var	3D.24	04.02.05.02.04.01	Opt	« Quality event data » from <qlty> chunk (see 7.4.2)	
Qlty End Of Modulation	UTF-16 String	Var	3D.25	04.02.05.02.05.01	Opt	« End modulation data » from <qlty> chunk (see 7.4.2)	
Qlty Quality Parameter	UTF-16 String	Var	3D.26	04.02.05.02.06.01	Opt	« Quality parameter data » from <qlty> chunk (see 7.4.2)	
Qlty Operator Comment	UTF-16 String	Var	3D.27	04.02.05.02.07.01	Opt	« Comments of operator » from <qlty> chunk (see 7.4.2)	
Qlty Cue Sheet	UTF-16 String	Var	3D.28	04.02.05.02.08.01	Opt	« Cue sheet data » from <qlty> chunk (see 7.4.2)	
Unknown BWF Chunks	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 7.4.4.	

NOTE1 All properties specified in Table A.5 are copied from the BWF <bext> chunk or <qlty> chunk or other chunk.

NOTE 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 can be further encoded as UTF-8 or shift-JIS in some geographic regions. When the Wave Audio Physical Descriptor is used to create BWF files, it is recommended that MXF decoders restrict the encoded values to those permitted in the particular region.

The key (UL) for this set is defined in Table A.6.

Table A.6 — Key for Wave Audio Physical Descriptor.

Byte No.	Description	Value (hex)	Meaning
1-13	Defined in the Structural Header Metadata Implementation clause of SMPTE ST 377-1		
14	Set Kind (1)	01h	Wave Audio Physical Descriptor
15	Set Kind (2)	50h	
16	Reserved	00h	Reserved

A.4 Unknown Chunk Set

Table A.7 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.

Table A.7 — Unknown Chunk Set.

	Item Name	Type	Len	Local Tag	UL Designator	Req ?	Meaning	Default
	Unknown Chunk	Set UL	16		See Table A.8	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 [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 [Specifies the reference to an overall modification]	
	Chunk ID	UInt32	4	4F.01	04.06.08.02	Req	The ID of the RIFF Chunk	
	Chunk Length	UInt32	4	4F.02	04.06.09.03	Req	The number of bytes of Chunk Data	
	Chunk Data	Stream	var	4F.03	04.07.04	Req	The bytes of the RIFF Chunk Data	

The key (UL) for this set is defined in Table A.8.

Table A.8 — Key for Unknown Chunk Set.

Byte No.	Description	Value (hex)	Meaning
1-13	Defined in the Structural Header Metadata Implementation clause of SMPTE ST 377-1		
14	Set Kind (1)	01h	Unknown Chunk 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 three sound clips. In each Source Clip 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 of the channels will be used.

The Wave Audio Essence Descriptor is used to describe the actual stored essence (e.g., properties such as Block Start Offset). 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.

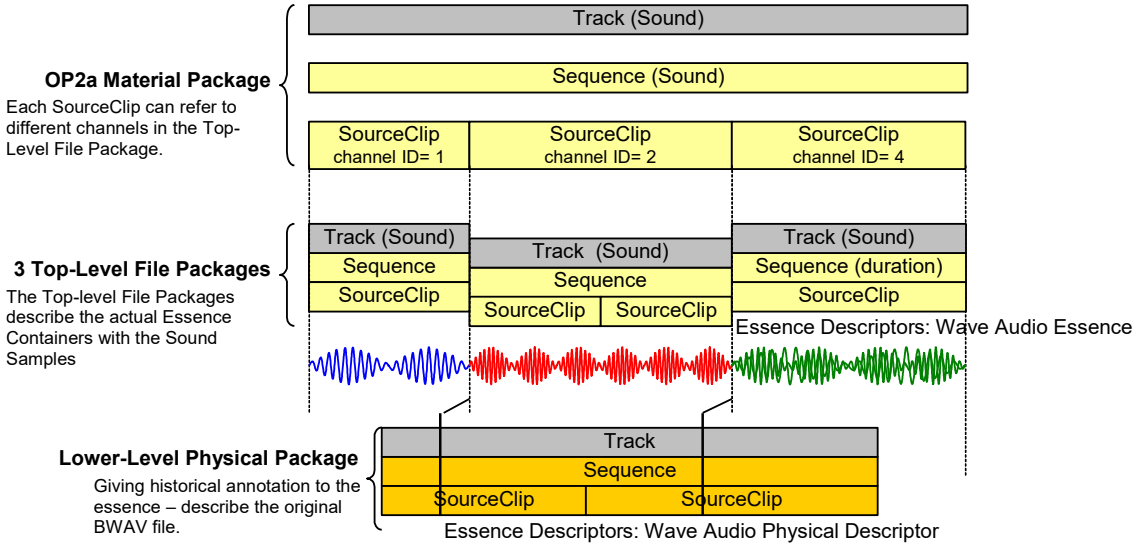

Figure B.1 — Package relationships and the use of the physical descriptor.

Annex C (Informative) Comparison with Other Standards

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

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

SMPTE ST 302 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, while efficient for MPEG systems, is not at all straightforward for byte-oriented processors. There are no provisions for carriage of clip-oriented metadata.

SMPTE ST 331 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 ST 331 itself does not carry clip-oriented metadata, although this is described fully in companion documents.

C.2 Comparison with SMPTE ST 337 / SMPTE ST 338 / SMPTE ST 339 and Related Standards

SMPTE ST 337 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 ST 337 data in an MXF essence container, without multiplexing.

SMPTE ST 338 defines a number of types of non-PCM data. This standard defines the mapping of SMPTE ST 338 data type definitions into the Sound Essence Coding property.

NOTE Some SMPTE ST 338 data types identify proprietary compressed audio data streams.

SMPTE ST 339 defines the format of a SMPTE ST 12-1 and SMPTE ST 309 time code packet within SMPTE ST 337 data_stream_number 7. This standard defines the carriage of SMPTE ST 339 time stamp data in the MXF time code track.

Annex D (Normative) Terminology Comparison with AES31-2

This standard was developed in parallel with AES31-2 and great care was taken to minimize any terminology differences. However, consistency with other standards in the MXF document suite was also taken into consideration, in particular with respect to data types. Occasionally the terminology differs from that used in AES31-2, even though the meanings of terms are identical.

Descriptions of terms normatively defined by AES31-2 are provided here for information. They are consistent with those given in the main body of this standard but use a slightly different vocabulary. Readers should therefore refer to AES31-2 for the complete descriptions.

D.1 Comparison of Data Type Definitions

Table D.1 — Mapping of BWFF to MXF data types.

MXF type	Equivalent AES31-2 type	AES31-2 definition
char	CHAR	8-bit signed integer, representing integer values from –128 to +127
UInt8	BYTE	8 bit unsigned integer, representing integer values from 0 to 255
Int16	INT	16-bit signed integer, representing integer values from –32768 to +32767
UInt16	WORD	16-bit unsigned integer, representing integer values from 0 to +65535
Int32	LONG	32-bit signed integer, representing integer values from –2,147,483,648 to +2,147,483,647
UInt32	DWORD	32-bit unsigned integer, representing integer values from 0 to +4,294,967,295

In AES31-2, multi-byte data types are little-endian.

D.2 Comparison of <fmt> Chunk Fields

Table D.2 — BWFF <fmt> chunk fields.

<fmt> field	AES (MXF) Type	AES31-2 description
wFormatTag	WORD (UInt16)	A number indicating the Wave format category of the file.
nChannels	WORD (UInt16)	The number of channels represented in the waveform data, such as 1 for mono or 2 for stereo.
nSamplesPerSec	DWORD (UInt32)	The sampling frequency, in samples per second, at which each channel should be reproduced.
nAvgBytesPerSec	DWORD (UInt32)	The average number of bytes per second at which the waveform data should be transferred. Playback software can estimate the buffer size using this value.
nBlockAlign	WORD (UInt16)	The block alignment in bytes of the waveform data. Playback software needs to process a multiple of nBlockAlign bytes of data at a time, so the value of nBlockAlign can be used for buffer alignment.
nBitsPerSample	WORD (UInt16)	The number of bits of data used to represent each audio sample of each channel. If there are multiple channels, the sample size shall be the same for each channel.

In AES31-2, nBitsPerSample is defined as a format-specific field that shall occur when wFormatTag is set to 0001h (WAVE_FORMAT_PCM), i.e., the Wave format category is Pulse Code Modulation (PCM) format. For other Wave format categories, it is not needed. If it is not present, then for the mapping of nBitsPerSample to MXF it should be assigned the value 0000h. This is consistent with the provisions of Clause 7.2.

D.3 Comparison of <bext> Chunk Fields

Table D.3 — BWFF <bext> chunk fields.

<bext> field	AES (MXF) Type	Readable by	AES31-2 description
Description	CHAR[256] (char[256])	Human	ASCII string, 256 characters or less, containing a description of the sequence. If data is not available or if the length of the string is less than 256 characters, the first unused character shall be a null character (00h). To help applications that only display a short description, a summary of the description should be contained in the first 64 characters. The last 192 characters may be used for details.
Originator	CHAR [32] (char[32])	Human	ASCII string, 32 characters or less, containing the name of the originator of the audio file. If data is not available or if the length of the string is less than 32 characters, the first unused character shall be a null character (00h).
OriginatorReference	CHAR[32] (char[32])	Human	ASCII string, 32 characters or less, containing a reference allocated by the originating organization. If data is not available or if the length of the string is less than 32 characters, the first unused character shall be a null character (00h).

<bext> field	AES (MXF) Type	Readable by	AES31-2 description
OriginationDate	CHAR[10] (char[10])	Human	ASCII string, 10 characters, containing the date of creation of the audio sequence. The date shall be represented as the year, month, and day of the Gregorian calendar. If data is unavailable, the default value shall be the origin of the modified Julian date (MJD), namely, 1858-11-17. **Format: CCYY-MM-DD** **CCYY** = 4 characters for century and year shall contain a value between 0000 and 9999 **-** = 1 character **MM** = 2 characters for month shall contain a value between 1 and 12 **-** = 1 character **DD** = 2 characters for day of month shall contain a value between 1 and 31 All components shall be present. Hyphen characters, "**-**", shall be used as separators within the date expression in compliance with ISO 8601. For compatibility with alternative implementations, reproducing equipment should also recognize the following separator characters: "_" underscore ":" colon " space "." period.
OriginationTime	CHAR[8] (char[8])	Human	ASCII string, 8 characters, containing the time of creation of the audio sequence in hours, minutes and seconds. If data is unavailable, the default value shall be 00:00:00. **Format: hh:mm:ss** **hh** = 2 characters shall contain a value between 00 and 23 if time given **:** = 1 character **mm** = 2 characters shall contain a value 00 - 59 if time given **:** = 1 character **ss** = 2 characters shall contain a value between 00 and 59 All components shall be present. Colon characters, ":", shall be used as separators within the time of day expression in compliance with ISO 8601. For compatibility with alternative implementations, reproducing equipment should also recognize the following characters: "_" underscore "-" hyphen " " space "." period.

<bext> field	AES (MXF) Type	Readable by	AES31-2 description
TimeReference(Low /High)	2 DWORDs (2 UInt32s)	Machine	This field shall contain the sample address count [time code] of the sequence. It is a 64-bit unsigned value which contains the sample count since midnight of the first sample in the audio data. The number of samples per second depends on the sample frequency which is defined in the field <nSamplesPerSec> from the <fmt-ck>. The default value is zero, corresponding to midnight.
Version	WORD (UInt16)	Machine	An unsigned binary number indicating the version of the BWF. For version 1 this value shall be 0001h.
UMID: UMID_0, … , UMID_63	64 BYTES (UInt8[64])	Machine	64 bytes containing an extended UMID to SMPTE ST 330. If a 32-byte basic UMID is used, the last 32 bytes shall be filled with zeros. If no UMID is available, the 64 bytes shall be filled with zeros. NOTE The length of the UMID is coded at the head of the UMID itself.
Reserved	CHAR[190] (char[190])	Machine	190 bytes reserved for extension. These 190 bytes shall be set to zero.
CodingHistory	CHAR [ ] (char[ ])	Human	A variable-size block of ASCII characters comprising 0 or more strings each terminated by <CR><LF>. The first unused character shall be a null character (00h). Each string shall contain a description of a coding process applied to the audio data. Each new coding application should add a new string with the appropriate information. See Annex B.8 of AES31-2.

ASCII is defined by AES31-2 to be a 7-bit character code compliant with ISO/IEC 646.

TimeReference is represented as 2 DWORDs, TimeReferenceLow and TimeReferenceHigh, as defined in 7.3.

Annex E (Normative) Channel ID Semantics

SMPTE ST 377-1 specifies an optional Channel ID property that may be used to select one or two channels out of N-channel essence described by a top-level file package, but leaves the semantics of such property to the specification applicable to the underlying audio essence. The following specifies the semantics of Channel ID for audio essence conforming to this specification.

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 ST 337, it is necessary to identify the source channel in greater detail, since each AES pair may contain multiple SMPTE ST 337 streams, within one or more subchannels, as provided by the underlying compression system. In this case, the Channel ID shall be interpreted as a structured number according to the following bit mapping, as specified in Table E.1:

Table E.1 — Structure of Channel ID property.

Bit numbers	Name	Meaning
31-24	Subchannel	1-based index of the Subchannel number within the SMPTE ST 337 AES pair 0 implies all Subchannels in a Stream
23	Stream Flag	1 if a specific Stream is selected by bits 22-16 0 implies all Streams in a SMPTE ST 337 AES Pair
22-16	Stream	When bit 23=1, 0-based data_stream_number within the SMPTE ST 337 AES pair Otherwise, 0
15-0	Subframe	Index of the source channel. For AES, this is the 1-based index of the subframe. For BWF, this is the 1-based index of the channel number.

NOTE 1 Channel ID in the range 1-65535 applies to AES or BWF essence.

NOTE 2 This encoding of bits 31-16 permits values of Channel ID in the range 1-65535 to have the same effect irrespective of whether the essence is compressed per SMPTE ST 337 or not.

Annex F (Normative)
Mapping Multiple SMPTE ST 339 Time Stamps

F.1 Association of SMPTE ST 337 data_stream_number to Timecode Tracks

If there are multiple Timecode Tracks in a file package, then the AES3 Audio Essence Descriptor :: Linked Timecode Track ID shall contain the value of the Track ID for the Timecode Track that was created from the time stamp data.

Applications that need to associate one of several Timecode Tracks with this essence shall set the AES3 Audio Essence Descriptor :: Linked Timecode Track ID to the value of the Timecode Track's Track ID property. The same descriptor may, in accordance with SMPTE ST 377-1, have its AES3 Audio Essence Descriptor :: Linked Track ID set to the value of the Sound Track's Track ID for this essence. This mechanism allows the association of:

- the essence container described by its Essence Descriptor;
- the Sound Track in the package for this essence;
- the Timecode Track in the package for this essence.

There may, of course, be multiple data streams within a single SMPTE ST 337 transport, each of which is identified by a unique data_stream_number. If multiple data streams are to be described in an MXF package, then each of them shall have a descriptor with the optional AES3 Audio Essence Descriptor :: SMPTE ST 337 Data Stream Number property set to the data_stream_number of the SMPTE ST 337 data stream being described. In this case, the SMPTE ST 339 mechanism is used to associate the data stream with its time stamp stream. The mechanism above associates the data stream with the MXF properties.

Annex G (Normative) Peak Envelope Chunk

The Peak Envelope Chunk (<levl> chunk) contains information about peak audio signal levels in the BWFF.

G.1 Generation of the Peak Envelope Data

The audio signal is divided into blocks of samples of constant size.

The samples of each channel are evaluated to find the peak points (extreme values). Separate peak points should be found for positive and negative samples, but alternatively the absolute value may be used to find a single peak point. All of the peak points are stored as unsigned values.

The peak points are rounded to one of two formats, either UInt8 or UInt16.

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

These peak frames are carried as the data in the <levl> chunk. The <levl> chunk starts with a header that contains information that allows the peak data to be interpreted.

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

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

G.2 Definition of the Peak Envelope Chunk

The <levl> chunk consists of a header followed by the data of the peak points.

The structure of the <levl> chunk shall be:

```
typedef struct peak_envelope{
   CHAR ckID[4];
   DWORD ckSize;
   DWORD dwVersion;
   DWORD dwFormat;
   DWORD dwPointsPerValue;
   DWORD dwBlockSize;
   DWORD dwPeakChannels;
   DWORD dwNumPeakFrames;
   DWORD dwPosPeakOfPeaks;
   DWORD dwOffsetToPeaks;
   CHAR strTimestamp[28];
   CHAR reserved[60];
   CHAR peak_envelope_data[ ]
} levl_chunk;
```

The data types CHAR and DWORD are defined in D.1.

The elements of the <levl> chunk are defined below:

ckID The 4 character array {"l", "e", "v", "l"}, the chunk identification.

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

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

dwFormat The format of the peak envelope data. Two formats are allowed:
1 = UInt8 for each peak point
2 = UInt16 for each peak point

NOTE A "negative" peak point is stored as a "positive" number.

dwPointsPerValue This denotes the number of peak points per peak value. This may be either 1 or 2.

dwPointsPerValue = 1:

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

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

dwPointsPerValue = 2:

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

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

dwPeakChannels The number of peak channels.

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

dwNumPeakFrames = (numAudioFrame + dwBlockSize)/dwBlockSize

or rounding up the following calculation:

dwNumPeakFrames = numAudioFrame / dwBlockSize,

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

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

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

dwOffsetToPeaks Offset of the peak data from the start of the header.

strTimeStamp Time stamp of the creation of the peak data. It is formatted as follows:

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

where YYYY is year, MM is month, DD is day, hh is hours, mm is minutes, ss is seconds, and uuu is milliseconds.

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

reserved Unused bytes set to 0x00

peak_envelope_data The peak point data

Annex H (Normative) Quality Chunk

The Quality Chunk (<qlty> chunk) contains a Quality Report and a Cue Sheet.

The Quality Report contains information describing all relevant events affecting the quality of the recorded sound signal in the wave data chunk. Each event, whether recognized by the operator or the computer, is listed with details of the type of event, exact time stamps, priority and event status. Overall quality parameters are also reported.

The Cue Sheet is a list of events marked with exact time stamps and further description of the sound signal, e.g., the beginning of an aria or the starting point of an important speech.

H.1 Syntax of the Quality Chunk

The Quality Chunk consists of strings of ASCII characters arranged in rows of up to 256 characters. (The definition of ASCII, in accordance with Clause D.3, is a 7-bit character code compliant with ISO/IEC 646.)

Each row should be terminated by <CR/LF> (ASCII 0Dh, 0Ah).

A row may contain one or more variable strings separated by commas (ASCII 2Bh).

Variable strings shall be in ASCII characters and should contain no commas.

Semicolons (ASCII 3Bh) should be used as separators within variable strings.

H.2 Definition of the Quality Chunk

The structure of the <qlty> chunk shall be:

```
Quality_chunk typedef struct {
   DWORD ckID;
   DWORD ckSize;
   BYTE ckData[ckSize];
}
typedef struct quality_chunk {
   DWORD FileSecurityReport;
   DWORD FileSecurityWave;
   CHAR BasicData[ ];
   CHAR StartModulation[ ];
   CHAR QualityEvent[ ];
   CHAR QualityParameter[ ];
   CHAR EndModulation[ ];
   CHAR QualityParameter[ ];
   CHAR OperatorComment[ ];
   CHAR CueSheet[ ];
} quality_chunk
```

The data types CHAR, BYTE and DWORD are defined in D.1.

The elements of the <qlty> chunk are defined below:

ckID	The 4 character array {"q", "l", "t", "y"}, the chunk identification.
ckSize	The size of the remainder of the chunk. (It shall not include the 8 bytes used by ckID and ckSize.)
ckData	The data of the chunk.
FileSecurityReport	FileSecurityCode of the <qlty> chunk. It is a 32-bit value that contains the checksum [0....231].
FileSecurityWave	FileSecurityCode of BWF Wave data. It is a 32-bit value that contains the checksum [0....231].
BasicData	Basic data of capturing
B=	ASCII string containing basic data about the sound material
Archive no. (AN):	Archive number (maximum 32 characters).
Title (TT):	Title / take of the sound data (maximum 256 characters).
Duration (TD):	10 ASCII characters containing the time duration of the sound sequence. Format: << hh:mm:ss:d >> Hours hh: 0...23 Minutes mm: 0...59 Seconds ss: 0...59 1/10 second d: 0...9
Date (DD):	10 ASCII characters containing the date of digitization. Format: << yyyy:mm:dd >> Year yyyy: 0000...9999 Month mm: 0...12 Day dd: 0...31
Operator (OP):	ASCII string (maximum 64 characters) containing the name of the person carrying out the digitizing operation.
Copying station (CS):	ASCII string (maximum 64 characters) containing the type and serial no. of the workstation used to create the file.
StartModulation	Start of modulation of the original recording
SM=	10 ASCII characters containing the starting time of the sound signal from the start of the file. Format: << hh:mm:ss:d >> Hours hh: 0...23 Minutes mm: 0...59 Seconds ss: 0...59 1/10 second d: 0...9
Sample count (SC):	Sample address code of the SM point from the start of the file. Format: << ########H >> 0H..... FFFFFFFFH

Comment (T):	ASCII string containing comments.		
QualityEvent	Information describing each quality event in the sound signal. One QualityEvent string is used for each event.		
Q=	ASCII string (maximum 256 characters) containing quality events.		
Event number (M):	Numbered mark originated manually by operator.		
	Format: << M### >> ###: 001...999		
Event number (A):	Numbered mark originated automatically by system.		
	Format: << A### >> ###: 001...999		
Priority (PRI):	Priority of the quality event		
	Format: << # >> #: 1 (LO)..... 5 (HI)		
Time stamp (TS):	10 ASCII characters containing the time stamp of the quality event from the start of the file.		
	Format: << hh:mm:ss:d >>		
	Hours hh: 0...23 Minutes mm: 0...59 Seconds ss: 0...59 1/10 second d: 0...9		
Event type (E):	ASCII string (maximum 16 characters) describing the type of event (e.g., "Click", "AnalogOver", "Transparency") or QualityParameters (defined below) exceeding limits, e.g., "QP:Azimuth:L-20.9smp".		
Status (S):	ASCII string (maximum 16 characters) containing the processing status of the event, e.g., "unclear", "checked", "restored", "deleted".		
Comment (T):	ASCII string containing comments.		
Sample count (SC):	Sample address code of the TS point from the start of the file.		
	Format: << ########H >> 0H..... FFFFFFFFH		
QualityParameter	Quality parameters describing the sound signal		
P=	ASCII string (maximum 256 characters) containing quality parameters.		
Parameters (QP):	MaxPeak:	-xx.x dBFSL;-yy.y dBFSR	[-99.9....-00.0]
	MeanLevel:	-xx.x dBFSL;-yy.y dBFSR	[-99.9....-00.0]
	Correlation:	±x.x	[-1.0.....+1.0]
	Dynamic: (Dynamic range)	xx.x dBL; yy.y dBR	[00.0.....99.9]
	ClippedSamples:	xxxx smpL; yyyy smpR	[0.....9999]
	SNR: (Signal-to-noise-ratio)	xx.x dBL; yy.y dBR	[00.0.....99.9]
	Bandwidth:	xxxxx HzL; yyyyy HzR	[0.....20000]
	Azimuth:	L±xx.x smp	[-99.9.....+99.9]
	Balance:	L±x.x dB	[-9.9.....+9.9]

	DC-Offset:	x.x %L; y.y %R	[0.0..... 9.9]
	Speech:	xx.x%	[0.0.....99.9]
	Stereo:	xx.x%	[0.0.....99.9]
	(L = left channel, R = right channel)		

Quality factor (QF):	Summary quality factor of the sound file [1.....5 (best), 0 = undefined].
Inspector(IN):	ASCII string (maximum 64 characters) containing the name of the person inspecting the sound file.
File status (FS):	ASCII character string describing the status "Ready for transmission?". [Y(es) / N(o) / U: File is ready / not ready / FS is undefined]
OperatorComment	Operator comments
T=	ASCII string (maximum 256 characters) containing comments.
EndModulation	End of modulation
EM=	10 ASCII characters containing the end of modulation time of the sound signal. Format: << hh:mm:ss:d >> Hours hh: 0...23 Minutes mm: 0...59 Seconds ss: 0...59 1/10 second d: 0...9
Sample count (SC):	Sample address code of the EM point. Format: << ########H >> 0H..... FFFFFFFFH
Comment (T):	ASCII string containing comments.
CueSheet	Cue sheet data
C=	ASCII string (maximum 256 characters) containing cue points.
Cue number (N):	Number of cue point automatically originated by the system. Format: << N### >> ###: 001...999
Time stamp (TS):	10 ASCII characters containing the time stamp of the cue point. Format: << hh:mm:ss:d >> Hours hh: 0...23 Minutes mm: 0...59 Seconds ss: 0...59 1/10 second d: 0...9
Text (T):	ASCII string containing describing comments of the cue point, e.g., "Beginning of an aria".
Sample count (SC):	Sample address code of the TS point. Format: << ########H >> 0H..... FFFFFFFFH

H.3 Example of a Quality Chunk

QualityReport in the Quality Chunk:

Line #

01 <FileSecurityReport>
02 <FileSecurityWave>
03 B=CS=QUADRIGA2.0; SN10012, OP=name_of_operator<CR/LF>
04 B=AN=archive_number, TT=title_of_sound<CR/LF>
05 B=DD= yyyy:mm:dd, TD=hh:mm:ss:d<CR/LF>
06 SM=00:00:04:5, T=tape noise changing to ambience, SC=34BC0H<CR/LF>
07 Q=A001, PRI=2, TS=00:01:04:0, E=Click, S=unclear, SC=2EE000H<CR/LF>
08 Q=A002, PRI=3, TS=00:12:10:3, E=DropOut, S=checked, SC=216E340H<CR/LF>
09 Q=A003, PRI=4, TS=00:14:23:0, E=Transparency, S=checked, SC=2781480H<CR/LF>
10 Q=M004, PRI=1, TS=00:18:23:1, E=PrintThrough, S=checked, SC=327EF40H<CR/LF>
11 Q=A005, PRI=4, TS=00:20:01:6, E=ClickOn, S=unclear, T=needs restoration, SC=3701400H<CR/LF>
12 Q=A006, PRI=5, TS=00:21:20:3, E=QP:Azimuth:L=-20.9smp, S=unclear, SC= 3A9B840H<CR/LF>
13 Q=A007, PRI=3, TS=00:21:44:7, E=AnalogOver, S=checked, SC=3BB9740H<CR/LF>
14 Q=A008, TS=00:22:11:7, E=ClickOff, SC=3BB9740H<CR/LF>
15 Q=A009, PRI=1, TS=00:28:04:0, E=DropOut, S=deleted, SC=4D16600H<CR/LF>
16 EM=00:39:01:5, T=fade-out of applause, SC=6B2F740H<CR/LF>
17 P=QP:MaxPeak:-2.1dBFSL;-2.8dBFSR<CR/LF>
18 P=QP:MeanLevel:-11.5dBFSL;-8.3dBFSR<CR/LF>
19 P=QP:Correlation:+0.8<CR/LF>
20 P=QP:Dynamic:51.4dBL;49.6dBR<CR/LF>
21 P=QP:ClippedSamples:0smpL;0smpR<CR/LF>
22 P=QP:SNR:32.3dBL;35.1dBR<CR/LF>
23 P=QP:Bandwidth:8687HzL;7943HzR<CR/LF>
24 P=QP:Azimuth:L-6.2smp<CR/LF>
25 P=QP:Balance L:+2.1dB<CR/LF>
26 P=QP:DC-Offset:0.0%L;0.0%R<CR/LF>
27 P=QP:Speech:64.2%<CR/LF>
28 P=QP:Stereo:89.3%<CR/LF>
29 P=QF=2<CR/LF>
30 P=IN=name_of_inspector<CR/LF>
31 P=FS=N<CR/LF>

CueSheet in the Quality Chunk:

Line #

32 C=N001, TS=00:17:02:5, T=beginning of speech, SC=2ECE6C0 H<CR/LF>
33 C=N002, TS=00:33:19:2, T=start of aria, SC=5B84200H<CR/LF>

Bibliography

SMPTE ST 12-1, Time and Control Code

SMPTE ST 302, Mapping of AES3 data into MPEG Transport Stream

SMPTE ST 309, Transmission of Date and Time Zone Information in Binary Groups of Time and Control Code

SMPTE ST 314:2005, Data Structure for DV-Based Audio, Data and Compressed Video Television — 25 and 50 Mb/s

SMPTE ST 2035, Audio Channel Assignments for Digital Television Recorders (DTRs)

SMPTE ST 323, Channel Assignments and Levels on Multichannel 5.1 Audio Media

SMPTE ST 330, Unique Material Identifier (UMID)

SMPTE ST 331, Element and Metadata Definitions for SDTI-CP

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

EBU Recommendation R 099, Unique Source Identifier (USID) for Use in the OriginatorReference Field of the Broadcast Wave Format

ISO/IEC 646, Information technology – ISO 7-Bit Coded Character Set for Information Exchange. Geneva CH: International Organization for Standardization.

ISO 8601, Data Elements and Interchange Formats — Information Interchange — Representation of Dates and Times

Recommendation ITU-R BS.1352, Broadcast Wave Format (BWF)