SMPTE PCD ST 2131:202x

SMPTE Public Committee Draft

Mapping Audio Definition Model (ADM) and other audio metadata RIFF Chunks to MXF

Table of Contents Page

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 File Formats and Systems.

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.

A conformant implementation according to this document is one that includes all mandatory provisions ("shall") and, if implemented, all recommended provisions ("should") as described. A conformant implementation need not implement optional provisions ("may") and need not implement them as described.

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; then formal languages; then figures; and then any other language forms.

Introduction

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

This document defines a generic mechanism for mapping RIFF Chunks containing audio metadata to MXF files (as specified in SMPTE ST 377-1). It also defines additional MXF support specifically for the mapping and labeling of ADM-described content.

Audio metadata RIFF Chunks can be synthesized for inclusion in MXF files, as well as being sourced from physical audio files such as Broadcast Wave 64-bit (BW64) files. The BW64 format is based on the WAVE audio file format, which in turn is based on the Resource Interchange File Format (RIFF). All such files are constructed from a collection of RIFF Chunks.

The BW64 format defines several RIFF Chunks for carrying metadata including Audio Definition Model (ADM) metadata. ADM metadata is specified in Recommendation ITU-R BS.2076 and provides an open, common metadata model for describing audio content beyond just fixed channel-based arrangements. For example, it supports channel-, object-, and scene-based audio to enable immersive and interactive experiences for broadcasting and cinema. In many workflows (particularly for post-produced audio content) the ADM metadata is carried in BW64 files along with PCM audio (as specified in Recommendation ITU-R BS.2088).

The approach taken in this document provides an alternative to the audio metadata mapping mechanisms defined in SMPTE ST 382 (mapping Broadcast Wave Format (BWF) and AES3 content to MXF) and SMPTE ST 2127-10 (mapping Metadata-Guided Audio (MGA) signals with Serial ADM (S-ADM) metadata to MXF). Each will be appropriate for different use cases.

The features of this approach include:

- Compatibility with any audio compression algorithm and any layout of channels across Sound Tracks
- Audio metadata can be added to MXF files without affecting MXF readers that do not support it
- There is no constraint on the overall structure of MXF files that are supported (for example, MXF files are permitted to conform to any Operational Pattern)
- Audio metadata RIFF chunks can be mapped unchanged to/from BW64 (or other RIFF) files making them easy to track and manage through workflows
- Support for RIFF Chunks containing very large amounts of metadata
- Support for any RIFF Chunk containing audio metadata (for example, some BW64 files rely on the inclusion of proprietary RIFF Chunks used by audio editing tools)

This enables several important use cases, such as:

- Augmenting (with ADM metadata) existing A/V MXF file structures used for television content delivery, mastering, archiving and playout workflows
- Using audio-only MXF files as an alternative to BW64 files in some audio editing and interchange workflows, with straightforward conversion between these formats
- Adding PCM audio accompanied by ADM metadata to Interoperable Master Format (IMF) Compositions (refer to SMPTE OV 2067-0)

Guidance for implementers of this document is provided in Annex A, notes on the design approach are provided in Annex B, and example scenarios are covered in Annex C. Example BW64 and MXF files are included with this document and are listed in Annex D.

Clause C.1 describes an example scenario (including an illustration, in Figure 1). It covers the mapping of ADM-described content to a simple MXF file with a single Essence Track, illustrating all the mechanisms defined in this document. In this example, ADM metadata is carried in an <axml> BW64 RIFF Chunk which is mapped to the MXF file. The PCM audio to which this ADM metadata relates is mapped to a Sound Track with three channels.

The body of the <axml> BW64 RIFF Chunk is carried in a Generic Stream Payload (in a Generic Stream Partition) which is owned by the `RIFFChunkDefinitionSubDescriptor` instance. This describes the mapped RIFF Chunk, most importantly identifying it as an <axml> BW64 RIFF Chunk through the `RIFFChunkID` element. An instance of the `RIFFChunkReferencesSubDescriptor` set identifies this RIFF Chunk as applicable to the Sound Track (this mechanism is necessary to support MXF files with more than one Sound Track). These mechanisms are defined in clause 5, with additional provisions relating to BW64 RIFF Chunks (such as <axml>) specified in clause 6.

ADM IDs are associated with all the channels of the Sound Track so that the PCM audio can be connected to the ADM metadata. This is done using an `ADM_CHNASubDescriptor` instance. This optional mechanism is defined in clause 7.

An `ADMAudioMetadataSubDescriptor` instance indicates that the RIFF Chunk contains ADM metadata, and provides a means to signal the Profiles and Levels to which the ADM metadata complies. This optional mechanism is defined in clause 8.

The optional Audio Labeling Framework for ADM-described Content is defined in clause 9 and is based on the Multichannel Audio (MCA) Labeling Framework defined in SMPTE ST 377-4. Each application is encouraged to define exactly how it will apply the framework; the intention is that it labels the soundfields that can be rendered from ADM-described content. In this example there are two `ADMSoundfieldGroupLabelSubDescriptor` instances, one for each of the two audioProgrammes defined by the ADM metadata. The utilization of the framework in this example is indicated by the `AudioLabelingFrameworkADMContent` label in the Wave Audio Essence Descriptor.

The Sound Track in this example complies with the optional Standard ADM Constraints configuration defined in clause 10 (assuming that the mapped <axml> BW64 RIFF Chunk consists of a “typical” ADM XML document). This configuration aims to improve interoperability by constraining certain core ADM-related aspects. It aims to reflect what would “typically” be expected of ADM-described content in an MXF file, placing constraints on aspects of both this document and Recommendation ITU-R BS.2088.

At the time of publication, no notice had been received by SMPTE claiming patent rights essential to the implementation of this Engineering Document. However, attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. SMPTE shall not be held responsible for identifying any or all such patent rights.

1 Scope

This document defines a mechanism for mapping Resource Interchange File Format (RIFF) Chunks containing audio metadata to MXF files to augment MXF Sound Tracks. It defines additional MXF support specifically for the mapping and labeling of content described by Audio Definition Model (ADM) metadata (Recommendation ITU-R BS.2076) which is often carried in Broadcast Wave 64-bit (BW64) RIFF files (Recommendation ITU-R BS.2088).

2 Normative References

The following standards 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 standards 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 377-1:2019, Material Exchange Format (MXF) – File Format Specification

SMPTE ST 377-4, MXF Multichannel Audio Labeling Framework

SMPTE ST 377-42, MCA Label Controlled Vocabulary

SMPTE ST 410, Material Exchange Format — Generic Stream Partition

SMPTE ST 429-17, XML Constraints

Recommendation ITU-R BS.2076, Audio definition model

Recommendation ITU-R BS.2088, Long-form file format for the international exchange of audio programme materials with metadata

W3C XML Signature Syntax and Processing

W3C Extensible Markup Language (XML) 1.0

ISO/IEC 646, Information Technology — ISO 7-Bit Coded Character Set for Information Interchange

NIST FIPS 180, Secure Hash Standard (SHS)

3 Terms and Definitions

For the purposes of this document, the terms and definitions given in SMPTE ST 410, SMPTE ST 377-1 and SMPTE ST 377-4 (with SMPTE ST 377-1 taking precedence in case of discrepancy) and the following apply:

3.1. RIFF Chunk

sequence of bytes consisting of (in order) `ckID`, `ckSize`, `ckData`

Note 1 to entry: `ckID` is an identifier indicating the purpose of the data payload and is comprised of four ISO 7-bit characters (occupying one byte per character); `ckSize` is a four-byte size indicator giving the size of the data payload; `ckData` is the data payload.

Note 2 to entry: A RIFF Chunk is sometimes followed by a padding byte – this is not a part of `ckData` and does not affect `ckSize`.

Note 3 to entry: Recommendation ITU-R BS.2088 provides a means to override the `ckSize` value for RIFF Chunks in BW64 files to permit payloads larger than the four-byte `ckSize` value would usually allow.

3.2. Resource Interchange File Format (RIFF)

generic file format consisting of a collection of RIFF Chunks

Note 1 to entry: RIFF is a long-established and simple file format without a canonical reference; it is also explained in Recommendation ITU-R BS.2088 and SMPTE ST 382.

3.3. BW64 RIFF Chunk

RIFF Chunk defined as part of the Broadcast Wave 64-bit file format specified in Recommendation ITU-R BS.2088

4 Overview: Applicability to different applications (Informative)

This document specifies a number of optional mechanisms that form a toolkit for adding audio metadata RIFF Chunks to MXF files. Each application will need to determine the appropriate mechanisms to use.

Applications that add ADM metadata to an MXF file using the RIFF Chunk mapping mechanism defined in this document are expected to meet any ADM-specific needs using the other mechanisms provided. The mechanisms adopted will be influenced by the ADM capabilities of the devices involved: the more sophisticated mechanisms might be more relevant to devices with additional ADM capabilities (such as the ability to parse the ADM metadata itself).

A range of applications are supported, including generic RIFF Chunk mapping (perhaps with no ADM metadata present), basic ADM support, and advanced ADM support. Table 1 gives several example applications and suggests the appropriate mechanisms for each one.

Table 1 – Suggested applicability of clauses of this document to example applications

	(5) Mapping RIFF Chunks to Generic Streams	(6) Additional provisions for mapping BW64 RIFF Chunks	(7) Utilizing the ADM_CHNASubDescriptor set	(8) Describing ADM metadata	(9) Audio Labeling Framework for ADM-described Content	(10) Configuration: Standard ADM Constraints
Map generic WAVE files (with arbitrary audio metadata RIFF Chunks) to MXF	•					
Map BW64 files (with and without ADM metadata) to MXF [No ADM parser]	•	•	•			
Map BW64 files (with ADM metadata) to MXF [With ADM parser or extra inputs]	•	•	•	•	•	
Store synthesized ADM-described content directly in MXF	•	•	•	•	•	•

5 Mapping RIFF Chunks to Generic Streams

5.1. Overview

Any number of RIFF Chunks may be mapped to an MXF file containing a Top-Level File Package with at least one Sound Track. These RIFF Chunks need not have unique `ckID` values.

For each RIFF Chunk that is mapped to the MXF file there shall be:

- One Generic Stream in accordance with clause 5.2
- One instance of the `RIFFChunkDefinitionSubDescriptor` set in accordance with clause 5.3
- One or more instances of the `RIFFChunkReferencesSubDescriptor` set that reference the RIFF Chunk (in accordance with clause 5.4)

The additional provisions given in clause 6 shall apply to the mapping of BW64 RIFF Chunks.

Applications should specify which RIFF Chunks are to be mapped to an MXF file.

NOTE The intention is that only RIFF Chunks containing audio-related metadata are mapped to an MXF file using the mechanisms defined in this document. The mapping of other RIFF Chunks is discouraged, for example: "structural" RIFF Chunks including "filler" or "junk" RIFF Chunks; those containing audio samples; RIFF Chunks relating to non-audio content.

5.2. Generic Stream

The provisions of SMPTE ST 410 shall apply.

The Generic Stream Payload shall consist of the RIFF Chunk’s `ckData` bytes. The RIFF Chunk’s `ckID` and `ckSize` values are not included. Any padding byte that might appear after the RIFF Chunk (for example in a BW64 file) is not included.

MXF encoders and decoders shall treat the Generic Stream Payload as an unstructured sequence of bytes. An application later seeking to interpret these bytes must consult the defining document for the RIFF Chunk for details (including any endian-ness considerations for embedded data structures).

The entire Generic Stream Payload shall be carried in a single Generic Stream Data Element in a single Generic Stream Partition (no repetitions are permitted).

In some other file formats (such as BW64) the physical location and order of some RIFF Chunks is constrained: these constraints shall not apply when RIFF Chunks are mapped to an MXF file.

The Generic Stream Data Element key shall be urn:smpte:ul:060e2b34.0101010c.0d010509.01000000

NOTE The maximum size of `ckData` that can be mapped to an MXF file will be determined only by the MXF mechanisms utilized. The `ckSize` element of the RIFF Chunk is discarded and so this does not limit the size of `ckData` that can be mapped to the Generic Stream Payload.

5.3. RIFFChunkDefinitionSubDescriptor

The `RIFFChunkDefinitionSubDescriptor` set shall:

- be as defined in Table 2 and Table 3
- be a subclass of the `SubDescriptor` set

Each instance of the `RIFFChunkDefinitionSubDescriptor` set shall:

- accurately describe the RIFF Chunk
- be referenced by the `SubDescriptors` element of the File Descriptor that is directly referenced by a Top-Level File Package

NOTE 1. This File Descriptor will be the Multiple Descriptor if the Top-Level File Package contains more than one Essence Track.

Table 2 – Group/Set and Element ULs for `RIFFChunkDefinitionSubDescriptor`

Item Name	Symbol	Item UL
RIFF Chunk Definition Sub-Descriptor	`RIFFChunkDefinitionSubDescriptor`	urn:smpte:ul:060e2b34.027f0101.0d010101.0101810d
RIFF Chunk Stream ID	`RIFFChunkStreamID`	urn:smpte:ul:060e2b34.0101010e.04020308.01000000
RIFF Chunk ID	`RIFFChunkID`	urn:smpte:ul:060e2b34.0101010e.04020308.02000000
RIFF Chunk UUID	`RIFFChunkUUID`	urn:smpte:ul:060e2b34.0101010e.04020308.03000000
RIFF Chunk Hash SHA-1	`RIFFChunkHashSHA1`	urn:smpte:ul:060e2b34.0101010e.04020308.04000000

NOTE 2. The value of byte 6 of the `RIFFChunkDefinitionSubDescriptor` UL to use in an MXF file will be determined according to SMPTE ST 377-1 (it will typically be 0x53).

Table 3 – Structure of `RIFFChunkDefinitionSubDescriptor`

Symbol	Type	Len	Req?	Meaning
`RIFFChunkDefinitionSubDescriptor`	Set UL	16	Req	RIFF Chunk Definition Sub-Descriptor
`Length`	BER length	var	Req	Set length
All items from the abstract `SubDescriptor` set as defined in SMPTE ST 377-1				
`RIFFChunkStreamID`	UInt32	4	Req	The Stream ID of the Generic Stream carrying the `ckData` bytes of the RIFF Chunk
`RIFFChunkID`	ISO7	4	Req	The `ckID` value of the RIFF Chunk
`RIFFChunkUUID`	UUID	16	Opt	A globally unique identifier for the RIFF Chunk
`RIFFChunkHashSHA1`	DataValue	20	Opt	The SHA-1 digest of the `ckData` bytes of the RIFF Chunk

NOTE 3. ISO7: a String consisting of ISO 7-bit characters (per ISO/IEC 646), as defined in SMPTE ST 377-1. The value of `RIFFChunkID` will consist of four characters and so will be four bytes in length.

NOTE 4. DataValue: an unstructured sequence of bytes. It is not an array or a batch and so does not have a header of eight bytes.

Additional `RIFFChunkStreamID` constraints:

- This linkage to a Generic Stream (by means of its Stream ID) shall confer ownership as defined in SMPTE ST 410. All other references to the same Stream ID shall constitute a secondary linkage mechanism without conferring ownership.

Additional `RIFFChunkID` constraints:

- Within a Top-Level File Package, instances of `RIFFChunkID` need not have unique values

NOTE 5. Multiple RIFF Chunks with the same `ckID` value can be defined within a Top-Level File Package, unless otherwise specified.

Additional RIFFChunkUUID constraints:

- Within a Top-Level File Package, no two instances of RIFFChunkUUID shall have the same value
- Any two RIFF Chunks described by the same RIFFChunkUUID shall have the same ckID value and the same ckData bytes

NOTE 6. A RIFF Chunk could, for example, appear in two different MXF files with the same RIFFChunkUUID. This could help with the management of RIFF Chunks through a workflow. An application would need to define how such UUID values are generated and utilized.

NOTE 7. As defined in SMPTE ST 377-1, UUIDs are generated as specified in IETF RFC 4122.

NOTE 8. This document places no constraints on the UUID version field value.

Additional RIFFChunkHashSHA1 constraints:

- The value of RIFFChunkHashSHA1 shall have a length of 20 bytes and shall be the SHA-1 message digest of the ckData bytes of the RIFF Chunk (the SHA-1 algorithm is defined in NIST FIPS 180)

NOTE 9. The element is not an array or a batch and so does not have a header of eight bytes. The element's value consists entirely of the binary output of the SHA-1 algorithm.

NOTE 10. RIFFChunkHashSHA1 could be used to ensure the data integrity of RIFF Chunks. Additionally, it could be used (in the absence of RIFFChunkUUID) to easily identify RIFF Chunks that have the same data, where those RIFF Chunks could be stored in multiple MXF files, BW64 files, metadata databases, etc.

5.4. RIFFChunkReferencesSubDescriptor

The RIFFChunkReferencesSubDescriptor set shall:

- be as defined in Table 4 and Table 5
- be a subclass of the SubDescriptor set

Each instance of the RIFFChunkReferencesSubDescriptor set shall:

- be referenced by the SubDescriptors element of the File Descriptor for a Sound Track in a Top-Level File Package
- reference all the RIFF Chunks that are applicable to that Sound Track

The File Descriptor for a Sound Track shall reference zero or one instances of the RIFFChunkReferencesSubDescriptor set.

NOTE 1. If the Top-Level File Package consists of just one Track (a Sound Track) then there will be only one File Descriptor in the Package. Therefore, this will reference instances of both the RIFFChunkDefinitionSubDescriptor set and the RIFFChunkReferencesSubDescriptor set.

Table 4 – Group/Set and Element ULs for `RIFFChunkReferencesSubDescriptor`

Item Name	Symbol	Item UL
RIFF Chunk References Sub-Descriptor	`RIFFChunkReferencesSubDescriptor`	urn:smpte:ul:060e2b34.027f0101.0d010101.01018110
RIFF Chunk Stream IDs Array	`RIFFChunkStreamIDsArray`	urn:smpte:ul:060e2b34.0101010e.04020308.06000000

NOTE 2. The value of byte 6 of the `RIFFChunkReferencesSubDescriptor` UL to use in an MXF file will be determined according to SMPTE ST 377-1 (it will typically be 0x53).

Table 5 – Structure of `RIFFChunkReferencesSubDescriptor`

Symbol	Type	Len	Req?	Meaning
`RIFFChunkReferencesSubDescriptor`	Set UL	16	Req	RIFF Chunk References Sub-Descriptor
`Length`	BER length	var	Req	Set length
All items from the abstract `SubDescriptor` set as defined in SMPTE ST 377-1				
`RIFFChunkStreamIDsArray`	UInt32Array	8 + 32n	Req	The Stream IDs for the applicable RIFF Chunks

Additional `RIFFChunkStreamIDsArray` constraints:

- `RIFFChunkStreamIDsArray` shall contain no duplicate values
- Each Stream ID in `RIFFChunkStreamIDsArray` shall identify an instance of the `RIFFChunkDefinitionSubDescriptor` set (by means of its `RIFFChunkStreamID` element). These instances shall all be within the same Top-Level File Package as the `RIFFChunkReferencesSubDescriptor` instance.

6 Additional provisions for mapping BW64 RIFF Chunks

6.1. Overview

The mapping of BW64 RIFF Chunks using the mechanism defined in clause 5 shall be in accordance with clauses 6.2 and 6.3.

6.2. General provisions

The mechanism defined in clause 5 shall not be used to map any of the following BW64 RIFF Chunks to an MXF file: <BW64>, <ds64>, <JUNK>, <chna>.

If a `RIFFChunkReferencesSubDescriptor` instance references at least one of the following BW64 RIFF Chunks:

<axml>, <bxml>, <sxml>

then the Rules for XML Chunks defined in Recommendation ITU-R BS.2088 shall apply to all the RIFF Chunks referenced by the `RIFFChunkReferencesSubDescriptor` instance, except that there shall not be a <chna> BW64 RIFF Chunk present.

Any ADM metadata carried in an <axml>, <bxml> or <sxml> BW64 RIFF Chunk shall not include the audioMXFLookUp ADM element.

NOTE 1. The `ADM_CHNASubDescriptor` set defined in clause 7 of this document is designed to perform the same functions as the <chna> BW64 RIFF Chunk and is used in its place (clause B.4 explains the reasoning behind the <chna> BW64 RIFF Chunk not being used directly in MXF files). However, its use is not mandated in order to allow more flexibility in MXF files than is possible in BW64 files. For example, the Sound Track essence might use a bitstream format that already associates ADM IDs with audio samples and so an `ADM_CHNASubDescriptor` instance would not be required.

NOTE 2. The <sxml> BW64 RIFF Chunk can be used to carry Serial ADM (S-ADM) metadata (Recommendation ITU-R BS.2125). However, the specialized MXF mapping defined in SMPTE ST 2127-10 might be more suitable.

6.3. Interpreting time-based metadata (including ADM metadata)

BW64 RIFF Chunks mapped to an MXF file (using the mechanism defined in clause 5) may contain metadata that references points on the timeline of the associated audio. Such time-based references shall be interpreted as references to points on the timeline of the associated Top-Level File Package Sound Track(s) with zero-valued references interpreted as references to the Zero Point (Position=0) of this Sound Track(s).

NOTE 1. This means that in most "typical" cases the situation is very simple: audio and its time-based metadata (such as ADM metadata in an <axml> BW64 RIFF Chunk) can be moved between BW64 files and MXF files without any conversion of either the audio or metadata.

NOTE 2. Care needs to be taken if the Sound Track has a non-zero Origin and/or if it references essence that is longer than the Package Duration (which gives the duration of the "playable" essence). This is most likely to occur if the MXF file also contains Pre-Charge and/or Roll-Out (video) content.

NOTE 3. Additionally, care needs to be taken if the Material Package(s) defines an output timeline for the MXF file which does not include all the "playable" essence of the Top-Level File Package Sound Track(s) (for example, in an OP3a MXF file).

NOTE 4. Clause B.3 provides further background on the differences between BW64 and MXF timelines.

7 Utilizing the ADM_CHNASubDescriptor set

7.1. Overview

Any Sound Track in a Top-Level File Package may utilize the `ADM_CHNASubDescriptor` set in accordance with clause 7.2.

NOTE The `ADM_CHNASubDescriptor` set is designed to perform the same functions as the <chna> BW64 RIFF Chunk: that is, it associates ADM IDs with the channels of the Sound Track. Clause B.4 explains the reasoning behind the <chna> BW64 RIFF Chunk not being used directly in MXF files. The `ADM_CHNASubDescriptor` set can be used by a Sound Track even if it does not reference any RIFF Chunks (and even if no RIFF Chunks are mapped to the MXF file at all), as discussed in clause C.3.

7.2. ADM_CHNASubDescriptor

The ADM_CHNASubDescriptor set shall:

- be as defined in Table 6 and Table 7
- be a subclass of the SubDescriptor set

Each instance of the ADM_CHNASubDescriptor set shall:

- be referenced by the SubDescriptors element of the File Descriptor for a Sound Track in a Top-Level File Package

The File Descriptor for a Sound Track shall reference zero or one instances of the ADM_CHNASubDescriptor set.

Table 6 – Group/Set and Element ULs for ADM_CHNASubDescriptor

Item Name	Symbol	Item UL
ADM CHNA Sub-Descriptor	ADM_CHNASubDescriptor	urn:smpte:ul:060e2b34.027f0101.0d010101.0101810e
Num Local Channels	NumLocalChannels	urn:smpte:ul:060e2b34.0101010e.04020309.01000000
Num ADM audioTrackUIDs	NumADMAudioTrackUIDs	urn:smpte:ul:060e2b34.0101010e.04020309.02000000
ADM Channel Mappings Array	ADMChannelMappingsArray	urn:smpte:ul:060e2b34.0101010e.04020309.03000000

NOTE 1. The value of byte 6 of the ADM_CHNASubDescriptor UL to use in an MXF file will be determined according to SMPTE ST 377-1 (it will typically be 0x53).

Table 7 – Structure of ADM_CHNASubDescriptor

Symbol	Type	Len	Req?	Meaning
ADM_CHNASubDescriptor	Set UL	16	Req	ADM CHNA Sub-Descriptor
Length	BER length	var	Req	Set length
All items from the abstract SubDescriptor set as defined in SMPTE ST 377-1				
NumLocalChannels	UInt16	2	Req	The number of MXF Sound Track channels referenced in ADMChannelMappingsArray
NumADMAudioTrackUIDs	UInt16	2	Req	The number of ADM audioTrackUID entities referenced in ADMChannelMappingsArray
ADMChannelMappingsArray	ADMChannelMappingStrongReferenceVector	8 + 16n	Req	Array of ADMChannelMapping instances

NOTE 2. ADMChannelMappingStrongReferenceVector: an array of Strong References to ADMChannelMapping instances.

Additional ADMChannelMappingsArray constraints:

- ADMChannelMappingsArray shall contain at least one entry

- Each `ADMChannelMapping` instance referenced by `ADMChannelMappingsArray` shall be in accordance with clause 7.3
- `ADMChannelMappingsArray` shall not reference any “null” or “unused” `ADMChannelMapping` instances (that is, in each `ADMChannelMapping` instance all elements shall have non-null values)

NOTE 3. By contrast, the <chna> BW64 RIFF Chunk allows unused entries to be included (filled with zero or null values) to facilitate later updating of the chunk without changing its size. In an MXF file, the KLV Fill item can be used to achieve a similar effect, if required.

7.3. ADMChannelMapping

The `ADMChannelMapping` set shall:

- be as defined in Table 8 and Table 9
- be a subclass of the `InterchangeObject` set

Table 8 – Group/Set and Element ULs for `ADMChannelMapping`

Item Name	Symbol	Item UL
ADM Channel Mapping	`ADMChannelMapping`	urn:smpte:ul:060e2b34.027f0101.0d010101.0101810f
Local Channel ID	`LocalChannelID`	urn:smpte:ul:060e2b34.0101010e.04020309.04000000
ADM audioTrackUID	`ADMAudioTrackUID`	urn:smpte:ul:060e2b34.0101010e.04020309.05000000
ADM audioTrackChannelFormatID	`ADMAudioTrackChannelFormatID`	urn:smpte:ul:060e2b34.0101010e.04020309.06000000
ADM audioPackFormatID	`ADMAudioPackFormatID`	urn:smpte:ul:060e2b34.0101010e.04020309.07000000

NOTE 1. The value of byte 6 of the `ADMChannelMapping` UL to use in an MXF file will be determined according to SMPTE ST 377-1 (it will typically be 0x53).

Table 9 – Structure of `ADMChannelMapping`

Symbol	**Type**	**Len**	**Req?**	**Meaning**
`ADMChannelMapping`	Set UL	16	Req	ADM Channel Mapping
`Length`	BER length	var	Req	Set length
All items from the abstract `InterchangeObject` set as defined in SMPTE ST 377-1				
`LocalChannelID`	UInt32	4	Req	The ID of the MXF Sound Track channel containing essence identified by the associated ADM audioTrackUID
`ADMAudioTrackUID`	UTF16String	var	Req	The UID of the associated ADM audioTrackUID, as defined in Rec. ITU-R BS.2076
`ADMAudioTrackChannelFormatID`	UTF16String	var	Req	The linked ADM audioTrackFormatID or ADM audioChannelFormatID value, as defined in Rec. ITU-R BS.2076
`ADMAudioPackFormatID`	UTF16String	var	Opt	The linked ADM audioPackFormatID value, as defined in Rec. ITU-R BS.2076

Additional `LocalChannelID` constraints:

- `LocalChannelID` shall use the same channel identifiers as the Channel IDs element defined in SMPTE ST 377-1:2019, clause B.23

NOTE 2. The MCA Channel ID element defined in SMPTE ST 377-4 also uses the same channel identifiers. The semantics of these identifiers are defined by essence container specifications. For example, channels of PCM audio mapped according to SMPTE ST 382 are identified using a 1-based index.

NOTE 3. `LocalChannelID` values will not always be unique because a channel can have multiple ADM audioTrackUID values associated with it.

NOTE 4. `LocalChannelID` performs the same function as the trackIndex element of the <chna> BW64 RIFF Chunk.

Additional `ADMAudioTrackUID` constraints:

- Each `ADMAudioTrackUID` value shall be unique within the `ADMChannelMapping` instances referenced by the `ADMChannelMappingsArray` element
- `ADMAudioTrackUID` shall be formatted as per the UID element of the <chna> BW64 RIFF Chunk

Additional `ADMAudioTrackChannelFormatID` constraints:

- ADMAudioTrackChannelFormatID shall be formatted as per the trackRef element of the <chna> BW64 RIFF Chunk

NOTE 5. This means a suffix will need to be added to an ADM audioChannelFormatID value if being used as the value for this element.

Additional ADMAudioPackFormatID constraints:

- ADMAudioPackFormatID shall be formatted as per the packRef element of the <chna> BW64 RIFF Chunk except that when an audioPackFormatID value is not required the ADMAudioPackFormatID element shall not be present (rather than being filled with null values)

7.4. Validating ADM_CHNASubDescriptor properties (Informative)

If the provisions relating to the ADM_CHNASubDescriptor set are correctly implemented, then the following statements will be true:

- NumLocalChannels > 0
- NumLocalChannels ≤ number of channels in the associated Sound Track
- NumADMAudioTrackUIDs > 0
- NumADMAudioTrackUIDs = number of entries in ADMChannelMappingsArray
- NumADMAudioTrackUIDs ≥ NumLocalChannels

8 Describing ADM metadata

8.1. Overview

Each RIFFChunkDefinitionSubDescriptor instance that describes a RIFF Chunk carrying ADM metadata shall be accompanied by zero or one instances of the ADMAudioMetadataSubDescriptor set in accordance with clause 8.2.

NOTE 1. If a RIFF Chunk is accompanied by an ADMAudioMetadataSubDescriptor instance then it is known that the RIFF Chunk is carrying ADM metadata. Nothing can be inferred from the absence of an accompanying ADMAudioMetadataSubDescriptor instance.

NOTE 2. Although use of the ADMAudioMetadataSubDescriptor set is optional in general, note that each RIFF Chunk referenced by an ADMSoundfieldGroupLabelSubDescriptor instance must also be referenced by an ADMAudioMetadataSubDescriptor instance (per clause 9).

8.2. ADMAudioMetadataSubDescriptor

The ADMAudioMetadataSubDescriptor set shall:

- be as defined in Table 10 and Table 11
- be a subclass of the SubDescriptor set

Each instance of the ADMAudioMetadataSubDescriptor set shall:

- reference a RIFF Chunk carrying ADM metadata
- accurately describe this ADM metadata

- be referenced by the SubDescriptors element of the same File Descriptor that references the accompanying RIFFChunkDefinitionSubDescriptor instance (that is, the RIFFChunkDefinitionSubDescriptor instance that describes the RIFF Chunk carrying the associated ADM metadata)

Table 10 – Group/Set and Element ULs for ADMAudioMetadataSubDescriptor

Item Name	Symbol	Item UL
ADM Audio Metadata Sub-Descriptor	ADMAudioMetadataSubDescriptor	urn:smpte:ul:060e2b34.027f0101.0d010101.01018111
RIFF Chunk Stream ID (link 1)	RIFFChunkStreamID_link1	urn:smpte:ul:060e2b34.0101010e.0402030a.01000000
ADM Profile and Level UL Batch	ADMProfileLevelULBatch	urn:smpte:ul:060e2b34.0101010e.0402030a.02000000

NOTE 1. The value of byte 6 of the ADMAudioMetadataSubDescriptor UL to use in an MXF file will be determined according to SMPTE ST 377-1 (it will typically be 0x53).

Table 11 – Structure of ADMAudioMetadataSubDescriptor

Symbol	Type	Len	Req?	Meaning
ADMAudioMetadataSubDescriptor	Set UL	16	Req	ADM Audio Metadata Sub-Descriptor
Length	BER length	var	Req	Set length
All items from the abstract SubDescriptor set as defined in SMPTE ST 377-1				
RIFFChunkStreamID_link1	UInt32	4	Req	The Stream ID for the RIFF Chunk carrying the associated ADM metadata
ADMProfileLevelULBatch	AUIDSet	8 + 16n	Opt	Batch of ULs, each representing a Profile and Level to which the associated ADM metadata complies

NOTE 2. AUIDSet: a batch of AUIDs. But in this case each AUID value must be a UL so ADMProfileLevelULBatch effectively has a type of “batch of UL”.

Additional RIFFChunkStreamID_link1 constraints:

- This Stream ID shall identify the accompanying RIFFChunkDefinitionSubDescriptor instance (by means of its RIFFChunkStreamID element)

Additional ADMProfileLevelULBatch constraints:

- Each entry in ADMProfileLevelULBatch shall be the UL of a label in accordance with clause 8.3

NOTE 3. The use of ADMProfileLevelULBatch is optional: if omitted then the ADMAudioMetadataSubDescriptor instance is merely indicating (by its presence) that the referenced RIFF Chunk is carrying ADM metadata.

8.3. ADMAudioMetadataProfilesLevels labels

Labels to represent ADM metadata Profiles and Levels should be defined under the Node specified in Table 12.

ADM metadata conforming to Recommendation ITU-R BS.2076 but not further constrained by any defined Profile or Level shall be identified by the label specified in Table 13.

NOTE The intention is to define one collection of labels that can also be used in the `SADMAudioMetadataSubDescriptor` defined in SMPTE ST 2127-10. While one central collection of labels would be preferable, the use of labels defined elsewhere (such as those defined by third parties) is not prohibited.

Table 12 – `ADMAudioMetadataProfilesLevels` Labels Node

Item Name	Symbol	Item UL
ADM Audio Metadata Profiles and Levels	`ADMAudioMetadataProfilesLevels`	urn:smpte:ul:060e2b34.0401010d.04020211.00000000

Table 13 – `ADM_ITU2076` label

Item Name	Symbol	Item UL
ADM (Recommendation ITU-R BS.2076)	`ADM_ITU2076`	urn:smpte:ul:060e2b34.0401010d.04020211.01010000

9 Audio Labeling Framework for ADM-described Content

9.1. Overview

A Top-Level File Package may utilize the Audio Labeling Framework for ADM-described Content (in accordance with clause 9.2) if it contains one or more RIFF Chunks carrying ADM metadata.

Applications should specify whether the framework is to be used and how its components are to be employed.

NOTE The framework allows a lot of flexibility in order to support many different applications. Guidance on applying this framework is given in clause 9.7.

9.2. General provisions

The Audio Labeling Framework for ADM-described Content shall be as defined in SMPTE ST 377-4 except that:

- The `ADMSoundfieldGroupLabelSubDescriptor` set (as defined in clause 9.3) shall replace the `SoundfieldGroupLabelSubDescriptor` set
- If the `GroupOfSoundfieldGroupsLabelSubDescriptor` set is required, its usage shall be in accordance with clause 9.5
- If the `AudioChannelLabelSubDescriptor` set is required, its usage should be defined in the related application standard

If the Audio Labeling Framework for ADM-described Content is utilized by a Top-Level File Package, then within this Top-Level File Package:

- There shall be one or more instances of the `ADMSoundfieldGroupLabelSubDescriptor` set
- Each RIFF Chunk carrying ADM metadata shall be referenced by one or more `ADMSoundfieldGroupLabelSubDescriptor` instances

- Each RIFF Chunk referenced by an `ADMSoundfieldGroupLabelSubDescriptor` instance shall also be referenced by an `ADMAudioMetadataSubDescriptor` instance
- The Controlled Vocabulary defined in SMPTE ST 377-41 should be adopted
- Utilization of the framework may be identified by the label defined in clause 9.6

9.3. ADMSoundfieldGroupLabelSubDescriptor

The `ADMSoundfieldGroupLabelSubDescriptor` set shall:

- be as defined in Table 14 and Table 15
- be a subclass of the `SoundfieldGroupLabelSubDescriptor` set

Each instance of the `ADMSoundfieldGroupLabelSubDescriptor` set shall:

- reference a RIFF Chunk carrying ADM metadata
- accurately describe aspects of this ADM metadata
- comply with all provisions defined in SMPTE ST 377-4 for the `SoundfieldGroupLabelSubDescriptor` set, except that each `ADMSoundfieldGroupLabelSubDescriptor` instance shall be referenced by zero or more `AudioChannelLabelSubDescriptor` instances (rather than "one or more")
- be referenced by the `SubDescriptors` element of a File Descriptor in a Top-Level File Package, where this File Descriptor shall be either:
    i. the Multiple Descriptor (if present)
    ii. the File Descriptor for a Sound Track, where this Sound Track shall reference the RIFF Chunk carrying the associated ADM metadata (using the `RIFFChunkReferencesSubDescriptor` in accordance with clause 5.4)

NOTE 1. Zero or more `ADMSoundfieldGroupLabelSubDescriptor` instances can be referenced from an eligible File Descriptor. For example, it might be convenient for the File Descriptor for a Sound Track to reference multiple `ADMSoundfieldGroupLabelSubDescriptor` instances. Refer to clause 9.7 for further discussion on the positioning of `ADMSoundfieldGroupLabelSubDescriptor` instances.

Table 14 – Group/Set and Element ULs for `ADMSoundfieldGroupLabelSubDescriptor`

Item Name	Symbol	Item UL
ADM Soundfield Group Label Sub-Descriptor	`ADMSoundfieldGroupLabelSubDescriptor`	urn:smpte:ul:060e2b34.027f0101.0d010101.01018112
RIFF Chunk Stream ID (link 2)	`RIFFChunkStreamID_link2`	urn:smpte:ul:060e2b34.0101010e.0402030b.01000000
ADM audioProgrammeID (SMPTE ST 2131)	`ADMAudioProgrammeID_ST2131`	urn:smpte:ul:060e2b34.0101010e.0402030b.02000000
ADM audioContentID (SMPTE ST 2131)	`ADMAudioContentID_ST2131`	urn:smpte:ul:060e2b34.0101010e.0402030b.03000000
ADM audioObjectID (SMPTE ST 2131)	`ADMAudioObjectID_ST2131`	urn:smpte:ul:060e2b34.0101010e.0402030b.04000000

NOTE 2. The value of byte 6 of the `ADMSoundfieldGroupLabelSubDescriptor` UL to use in an MXF file will be determined according to SMPTE ST 377-1 (it will typically be 0x53).

Table 15 – Structure of `ADMSoundfieldGroupLabelSubDescriptor`

Symbol	Type	Len	Req?	Meaning
`ADMSoundfieldGroupLabelSubDescriptor`	Set UL	16	Req	ADM Soundfield Group Label Sub-Descriptor
`Length`	BER length	var	Req	Set length
All items from the `SoundfieldGroupLabelSubDescriptor` set as defined in SMPTE ST 377-4				
`RIFFChunkStreamID_link2`	UInt32	4	Req	The Stream ID for the RIFF Chunk carrying the associated ADM metadata
`ADMAudioProgrammeID_ST2131`	UTF16String	var	Opt	audioProgrammeID of the associated ADM audioProgramme, as defined in Rec. ITU-R BS.2076
`ADMAudioContentID_ST2131`	UTF16String	var	Opt	audioContentID of the associated ADM audioContent, as defined in Rec. ITU-R BS.2076
`ADMAudioObjectID_ST2131`	UTF16String	var	Opt	audioObjectID of the associated ADM audioObject, as defined in Rec. ITU-R BS.2076

Additional `RIFFChunkStreamID_link2` constraints:

- This Stream ID shall identify an instance of the `RIFFChunkDefinitionSubDescriptor` set (by means of its `RIFFChunkStreamID` element) and an instance of the `ADMAudioMetadataSubDescriptor` set (by means of its `RIFFChunkStreamID_link1` element). These instances shall both be within the same Top-Level File Package as the `ADMSoundfieldGroupLabelSubDescriptor` instance.

The MCA Label Dictionary ID, MCA Tag Symbol and MCA Tag Name elements shall be populated in accordance with one of the following three options:

a) populated in accordance with clause 9.4

b) populated by a Soundfield Group Label in the MCA Label Controlled Vocabulary (defined by SMTPE ST 377-42) mapped in accordance with Table 16

c) populated in accordance with another scheme

If option (a) is not adopted then option (b) should be adopted.

NOTE 3. In Option (b) the intention is that Soundfield Group Labels are mapped in the same way as for IMF Audio Track Files. The provisions are the same as defined in SMPTE ST 2067-2:2020 Table 8 except that Option (b) references SMPTE ST 377-42 instead of SMPTE ST 2067-8. The MCA Label Controlled Vocabulary defined by SMPTE ST 377-42 includes all of the MCA Label values defined by SMPTE ST 2067-8.

Table 16 – SMPTE ST 377-42 Soundfield Group Labels usage

SMPTE ST 377-4 element	SMPTE ST 377-42 parameter mapping
MCA Label Dictionary ID	Item UL
MCA Tag Symbol	Symbol prepended with the string ‘sg’
MCA Tag Name	MCA Tag Name

9.4. ADM Soundfield Group Label

Any `ADMSoundfieldGroupLabelSubDescriptor` instance may be labeled in accordance with Table 17.

NOTE The use of this ADM Soundfield Group Label is optional. If used, the elements indicated in Table 17 are to be populated as indicated: for example, the MCA Tag Symbol will be ‘ADM’ and not ‘sgADM’.

Table 17 – ADM Soundfield Group Label usage

SMPTE ST 377-4 element	Value
MCA Label Dictionary ID	`ADMSoundfield` UL as defined in Table 18
MCA Tag Symbol	‘ADM’
MCA Tag Name	‘ADM’

Table 18 – `ADMSoundfield` label

Item Name	Symbol	Item UL
ADM Soundfield	`ADMSoundfield`	urn:smpte:ul:060e2b34.0401010d.03020223.00000000

9.5. Usage of the GroupOfSoundfieldGroupsLabelSubDescriptor

For any `GroupOfSoundfieldGroupsLabelSubDescriptor` instance referenced by an instance of the `ADMSoundfieldGroupLabelSubDescriptor` set:

- the MCA Label Dictionary ID, MCA Tag Symbol and MCA Tag Name elements should be populated by a Group of Soundfield Groups Label in the MCA Label Controlled Vocabulary (defined by SMTPE ST 377-42) mapped in accordance with Table 19

NOTE As for option (b) in clause 9.3, the intention is that Group of Soundfield Groups Labels are mapped in the same way as for IMF Audio Track Files.

Table 19 – SMPTE ST 377-42 Group of Soundfield Groups Labels usage

SMPTE ST 377-4 element	SMPTE ST 377-42 parameter mapping
MCA Label Dictionary ID	Item UL
MCA Tag Symbol	Symbol prepended with the string ‘gg’
MCA Tag Name	MCA Tag Name

9.6. Framework Identification Label

Utilization of the Audio Labeling Framework for ADM-described Content may be identified by the label defined in Table 20.

NOTE For example, this label could be used in the Channel Assignment element of the Wave Audio Essence Descriptor (defined in SMPTE ST 382) for applicable Sound Tracks.

Table 20 – `AudioLabelingFrameworkADMContent` label

Item Name	Symbol	Item UL
Audio Labeling Framework for ADM-described Content	`AudioLabelingFrameworkADMContent`	urn:smpte:ul:060e2b34.0401010d.04020210.05010000

9.7. Applying the framework to applications (Informative)

The framework allows a lot of flexibility to support many different applications. Ideally each application would specify whether the framework is to be used and how its components are to be employed.

The intention is that the framework enables the labeling of the soundfields that can be rendered from ADM-described content, regardless of the type of audio (such as channel-, object-, or scene-based audio). In this way, the choices available in the ADM-described content can be exposed in the MXF Header Metadata. For example:

- There could be one instance of the `ADMSoundfieldGroupLabelSubDescriptor` set for each playback configuration available in the ADM-described content
- Each instance might list the ID of one ADM entity (audioProgramme, audioContent or audioObject), choosing the highest-level ADM entity that is applicable to that playback configuration. This would aid in navigating the associated ADM metadata.
- Each instance could be enriched with the metadata elements defined by SMPTE ST 377-4. This would be particularly important if multiple instances list the same ADM entity ID. For example, there might be two playback configurations associated with one audioProgramme, each with dialog in a different language, and so the RFC5646 Spoken Language element is populated.

Some additional considerations for applications:

- The option used for populating MCA Label Dictionary ID, MCA Tag Symbol and MCA Tag Name elements. Option (b) in clause 9.3 might be useful for channel-based audio that is described by ADM metadata.

- How ADMSoundfieldGroupLabelSubDescriptor instances are positioned: within the Multiple Descriptor or within the File Descriptor for a Sound Track. The difference will be purely cosmetic unless an application specifies a means of interpreting the position of an ADMSoundfieldGroupLabelSubDescriptor instance: this document does not require the audio samples of a Sound Track to be related to the soundfields described by any ADMSoundfieldGroupLabelSubDescriptor instances it references. Refer to clause C.2 for further discussion.
- Whether the original (unaltered) labeling framework defined in SMPTE ST 377-4 is also to be used in the same MXF file. This might be relevant if an MXF file contains a mixture of ADM-described and non-ADM audio content.

10 Configuration: Standard ADM Constraints

10.1. Overview

A Sound Track in a Top-Level File Package may comply with the Standard ADM Constraints configuration consisting of the provisions specified in clause 10.2.

NOTE This configuration aims to improve interoperability by constraining certain core ADM-related aspects. It aims to reflect what is “typically” expected of ADM-described content in an MXF file, placing constraints on aspects of both this document and Recommendation ITU-R BS.2088.

10.2. Constraints

The SubDescriptors element of the File Descriptor for the Sound Track shall reference:

- one instance of the ADM_CHNASubDescriptor set
- one instance of the RIFFChunkReferencesSubDescriptor set

and may reference instances of other SubDescriptor sets.

This RIFFChunkReferencesSubDescriptor instance shall reference the following RIFF Chunks, mapped according to clause 5:

- zero <bxml> BW64 RIFF Chunks
- zero <sxml> BW64 RIFF Chunks
- one <axml> BW64 RIFF Chunk

and may reference other RIFF Chunks.

The xmlData element of this <axml> BW64 RIFF Chunk shall consist of a well-formed and valid XML document as defined by W3C Extensible Markup Language (XML) 1.0 and as constrained by SMPTE ST 429-17.

This XML document shall contain exactly one audioFormatExtended ADM element which may be the root element.

NOTE 1. The <axml> BW64 RIFF Chunk will therefore contain just one top-level (root) XML element. This might be an audioFormatExtended XML element (containing ADM metadata). Alternatively, the audioFormatExtended XML element can be inside a larger XML document, such as an EBUCore XML document (as specified in EBU Tech 3293).

NOTE 2. By contrast, in a BW64 file the <axml> BW64 RIFF Chunk is allowed to contain multiple top-level XML elements (that is, multiple XML fragments are permitted).

10.3. Aspects not constrained (Informative)

There are many aspects left unconstrained by this configuration, such as:

- Whether other RIFF Chunks are referenced
- The structure of the XML document in the <axml> BW64 RIFF Chunk
- Any limitations on the ADM metadata, for example whether it complies with a particular profile
- Whether the `ADMAudioMetadataSubDescriptor` set is utilized (clause 8)
- Whether the Audio Labeling Framework for ADM-described Content (clause 9) is utilized (and which of its optional aspects are utilized)
- Whether the Sound Track contains all/subset/superset of the essence required for the entities defined in the ADM metadata, and whether all the relevant audio channels have ADM IDs assigned (by the `ADM_CHNASubDescriptor` instance)
- The Sound Track codec and audio sampling properties
- The structure (for example, Operational Pattern) of the MXF file containing the Sound Track

Annex A (Informative) Guidance for implementers

A.1 RF64 files

The BW64 file format specified in Recommendation ITU-R BS.2088 supersedes the RF64 file format (also known as the MBWF file format) specified in EBU Tech 3306. The RF64 file format was a major source for the design of the BW64 file format, and so they are very similar.

One difference between the two is the string of characters used at the start of a file. Both begin with a sequence (four bytes) of ISO 7-bit characters. A BW64 file begins with “BW64” whereas a RF64 file begins with “RF64”.

However, not all implementations have been updated to implement this change at the start of the file. As a result, files exist which are compatible with the BW64 file format except that they begin with the string “RF64”.

Implementers of this document are encouraged to:

- be tolerant of this discrepancy when handling BW64 files
- adhere strictly to Recommendation ITU-R BS.2088 when creating new BW64 files

Note that a BW64 file that is smaller than 4 GB will typically be created so as to enable increased backwards compatibility with WAVE files, in which case the first four bytes of the file will be “RIFF”.

A.2 Translating property values during MXF/BW64 conversion

If converting between MXF files and BW64 (or other WAVE) files, care needs to be taken that any property values being translated are handled correctly.

One important difference is that, in general, MXF files use big-endian encoding whereas BW64 files use little-endian encoding. For example, it might be necessary to create an instance of the `ADM_CHNASubDescriptor` set from a <chna> BW64 RIFF Chunk in a BW64 file, with the numTracks value being mapped to the `NumLocalChannels` element in the MXF file. Both are UInt16 values but the difference in endian-ness between the two file formats means that the bytes cannot be directly copied from the BW64 file to the MXF file.

However, note that (as explained in clause 5.2) when a RIFF Chunk’s `ckData` bytes are used to populate a Generic Stream Payload in an MXF file, the bytes are directly copied without any interpretation or translation. For example:

- There is no consideration of endian-ness: the payload is carried intact
- If the payload itself is constructed of RIFF Chunks (for example, in the case of some <LIST> RIFF Chunks) these are not unpacked

Some BW64 file properties use strings of ISO 7-bit characters. While some of the MXF properties are also strings of ISO 7-bit characters, others are UTF-16 strings. For example, if a packRef element of the <chna> BW64 RIFF Chunk is being mapped to an `ADMAudioPackFormatID` element in the MXF file, the value needs to be converted from a ISO 7-bit string to a UTF-16 string.

A.3 Editing or updating MXF files

Recommendation ITU-R BS.2088 gives some consideration to how BW64 files can be created dynamically, or later updated, for example by allowing the <chna> BW64 RIFF Chunk to contain “null” entries which can be overwritten later as required.

This document does not give any special consideration to these matters in MXF files. However, if similar capabilities are required then standard MXF approaches could be considered, such as: use of the KLV Fill item to reserve space to allow a piece of Header Metadata to grow; positioning a Generic Stream Partition close to the end of the MXF file if it contains RIFF Chunk data that might need to be expanded.

A.4 Variations permitted by this document

This document allows flexibility in order to support a variety of use cases. This means that implementations could encounter MXF files exhibiting numerous variations.

Some of these variations are highlighted by the Standard ADM Constraints configuration specified in clause 10, which lists several aspects that are constrained (and several that are not) if this configuration is utilized. The examples in Annex C also explore some of the possible variations.

Some of the other possible variations to be especially aware of include:

- `RIFFChunkDefinitionSubDescriptor` instances (and any accompanying `ADMAudioMetadataSubDescriptor` instances) will either be in the Sound Essence Descriptor (in the case that there is only one Essence Track and it is a Sound Track) or in the Multiple Descriptor (in the case that there is more than one Essence Track)
- `ADMSoundfieldGroupLabelSubDescriptor` instances will either be in the Sound Essence Descriptor or in the Multiple Descriptor, as determined by application requirements or use case
- A RIFF Chunk containing ADM metadata (such as the <axml> BW64 RIFF Chunk):
    1. might be referenced from multiple Sound Tracks. For example, ADM-described audio might be split-up into: stereo and 5.1 Sound Tracks; mono Sound Tracks.
    2. might be referenced from a Sound Track that does not have an associated `ADM_CHNASubDescriptor` instance (or vice versa)

Annex B (Informative) Notes on design approach

B.1 Conceptual role of the Top-Level File Package and the Material Package

Consider some ADM-described content that is mapped from a BW64 PCM+ADM file to an MXF file:

- the audio samples are mapped to regular Sound Tracks in a Top-Level File Package
- the ADM metadata (for example, in an <axml> BW64 RIFF Chunk) is mapped to a Generic Stream that is owned by the same Top-Level File Package
- a Material Package is created containing Sound Tracks that reference the Sound Tracks in the Top-Level File Package

In this way, the ADM-described content is owned by the Top-Level File Package in the same way as most other essence. However, straightforward playback of the MXF file (as specified by the Material Package) will result in the direct playback of the audio samples of the ADM-described content. While this is very convenient for monitoring and debugging, this is not a rendition of the ADM-described content. For that, an ADM renderer is required along with the information needed to control it (such as: whether a subset of the ADM metadata is be used; the arrangement of speakers to render for; etc.)

So, the approach taken is practical and straightforward, but does mean that the Material Package is not truly and completely defining the rendered output of the MXF file (which is its conceptual purpose). The only way to achieve this would be to deconstruct the ADM metadata and map all the ADM concepts of audio storage and content etc to the equivalent MXF concepts (probably across multiple packages including Lower-Level Source Packages). Even if possible, this would be highly impractical and be counter to the objective of allowing ADM-content to move seamlessly between workflows consisting of a mix of MXF and BW64 files.

B.2 Use of the Multiple Descriptor for the RIFFChunkDefinitionSubDescriptor

It is unusual for the Multiple Descriptor to reference instances of `SubDescriptor` sets. However, there are other examples such as SMPTE ST 379-2:2010 which requires users to: "add a `ContainerConstraintSubDescriptor` to the `GenericDescriptor::SubDescriptors` property of the top-most File Descriptor that describes the essence container". This File Descriptor will be the Multiple Descriptor when there are multiple Tracks.

Each `RIFFChunkDefinitionSubDescriptor` instance "owns" the associated Generic Stream (per SMPTE ST 410) and so cannot be duplicated. Therefore, when there are multiple Tracks, the Multiple Descriptor is a logical place to position it and this allows it to be found without searching through multiple (unrelated) File Descriptors.

B.3 Comparing BW64 and MXF timelines

In a BW64 file the first stored audio sample is placed at the zero point of the audio timeline; and the timeline has a duration equal to the duration of the stored audio samples. There is only one timeline for the audio and it covers its storage and playout. As such, all of the audio is "playable".

The provisions of clause 6.3 are necessary because of the extra layers of abstraction that MXF has in comparison to BW64, as described by Material Packages and File Packages (and their Tracks).

The provisions mean that time-based metadata in a mapped BW64 RIFF Chunk is considered to reference the "playable" essence of a Top-Level File Package Sound Track (the duration of this "playable" essence is given by the Package Duration of the Package). This is simple in most cases: normally the Sound Track merely reflects the associated audio essence stored in the MXF file; as such, it is all “playable” just as it is in a BW64 file.

B.4 The <chna> BW64 RIFF Chunk is treated differently

The <chna> BW64 RIFF Chunk is not mapped directly to the MXF file in the same way as other RIFF Chunks. Instead, the `ADM_CHNASubDescriptor` set is used to perform the same function. Some of the motivations include:

- The <chna> BW64 RIFF Chunk references the “tracks” of audio in the BW64 file, whereas in an MXF file the channels of the Sound Tracks need to be referenced. Not only is the terminology different, but the audio samples might be rearranged when ported to an MXF file (with the audio samples split across multiple Sound Tracks) which would require the values in the RIFF Chunk to be updated anyway.
- Recommendation ITU-R BS.2088 states that an additional function of the <chna> BW64 RIFF Chunk is that it allows faster access to ADM IDs. In an MXF file, such “faster access” is achieved by these ADM IDs being in the Header Metadata rather than in a Generic Stream. This makes them more accessible in various scenarios: for example, it means they will appear in the File Descriptor details included in an IMF Composition Playlist (CPL).

B.5 Relationship to SMPTE ST 2127-10

The Audio Labeling Framework for ADM-described Content defined in clause 9 is based on the audio labeling provisions of SMPTE ST 2127-10 (and, in turn, SMPTE ST 2127-1), to aid with content in MXF files being converted between ADM and S-ADM.

Additionally, the mechanisms defined in this document are designed to be flexible so that they can be combined with those defined in SMPTE ST 2127-1 and SMPTE ST 2127-10, if required. For example, S-ADM-described content might be mapped to an MXF file in accordance with SMPTE ST 2127-10 and then augmented by some audio metadata RIFF Chunks.

Annex C (Informative) Example scenarios

C.1 BW64 PCM+ADM file to MXF file using SMPTE ST 382 and SMPTE ST 2131

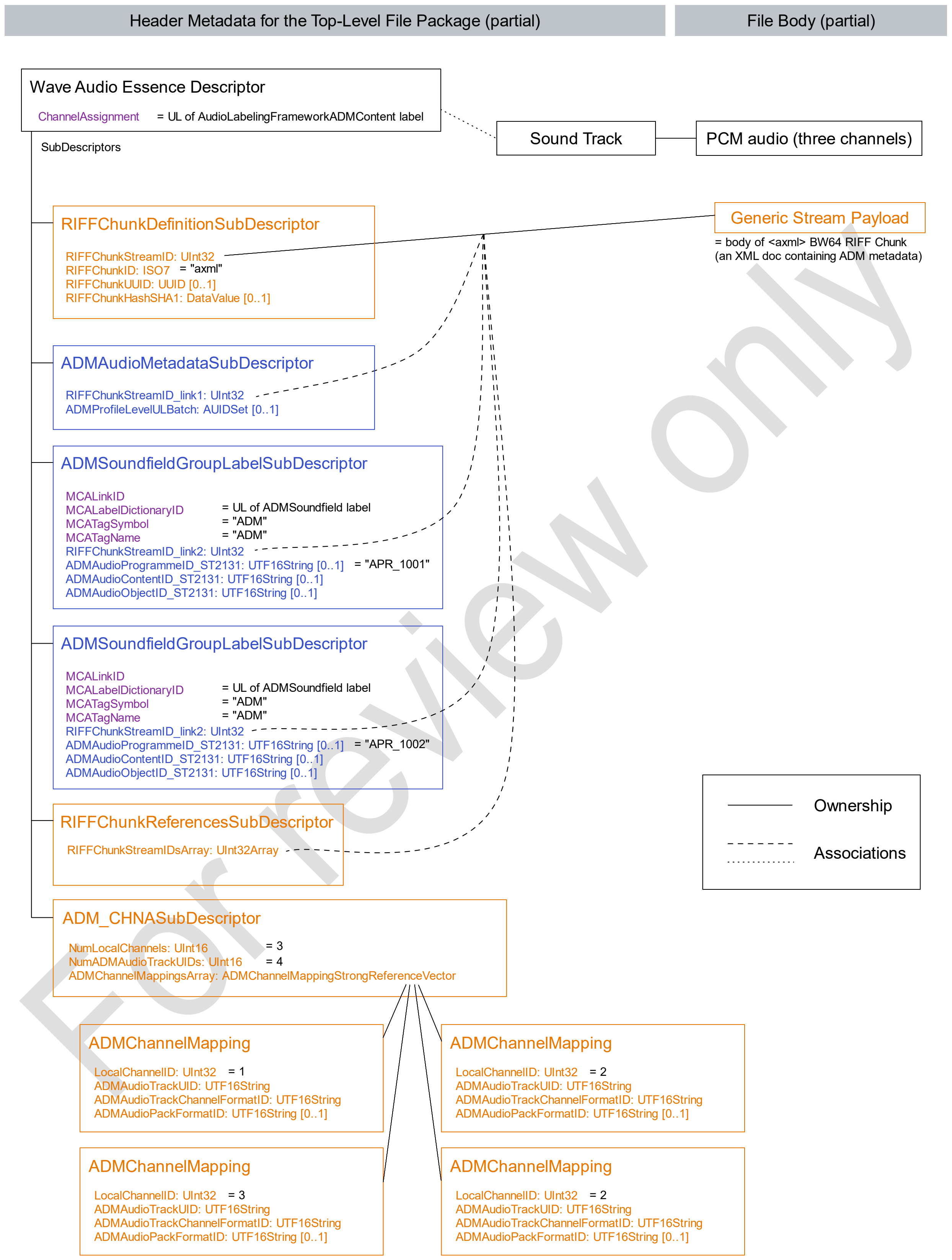
The Introduction describes an example scenario that is illustrated in Figure 1. It describes the mapping of ADM-described content to a simple MXF file with a single Essence Track, illustrating all the mechanisms defined in SMPTE ST 2131. That example could be the result of directly writing synthesized ADM-content into an MXF file. It could also be the result of creating an MXF file from a BW64 PCM+ADM file – it is this scenario that is considered here.

In this example, the MXF file is OP1a and has a single Essence Track (this is a Sound Track). The PCM audio samples are wrapped and described (using the Wave Audio Essence Descriptor) as defined in SMPTE ST 382. Each “track” of audio in the BW64 file becomes a channel in the single MXF Sound Track.

The <axml> BW64 RIFF Chunk in the BW64 file is mapped to the MXF file in accordance with clause 5, resulting in the Generic Stream Payload, and the `RIFFChunkDefinitionSubDescriptor` and `RIFFChunkReferencesSubDescriptor` instances. Any further RIFF Chunks in the BW64 file could also be mapped to the MXF file in the same way (subject to the constraints imposed by clause 6). The <chna> BW64 RIFF Chunk in the BW64 file is translated to an `ADM_CHNASubDescriptor` instance in accordance with clause 7. Note that in this example, the ADM-described content includes two non-overlapping audio objects which share the second audio channel; as a result, there are four `ADMChannelMapping` instances even though there are only three channels in the Sound Track.

These are all the core mechanisms of SMPTE ST 2131: they are illustrated in orange in Figure 1. In many cases these mechanisms will be sufficient when creating an MXF file from a BW64 PCM+ADM file. If it is known that all the BW64 RIFF Chunks in the source BW64 file already comply with the provisions of clause 6.2 (which is almost certainly the case for all compliant, real world BW64 files) then no further action is needed. Importantly, the contents of the <axml> BW64 RIFF Chunk (which could legitimately not contain any ADM metadata) does not need to be inspected.

An ADM-aware application might choose to add additional Header Metadata using the other mechanisms defined in SMPTE ST 2131 (shown in blue and purple in Figure 1, with the blue structures defined in this document). In this example, an `ADMAudioMetadataSubDescriptor` instance is added (in accordance with clause 8) because the <axml> BW64 RIFF Chunk is found to contain ADM metadata. The ADM metadata is parsed and found to define two audioProgrammes, and so the Audio Labeling Framework for ADM-described Content is utilized (in accordance with clause 9) to identify and label these using two `ADMSoundfieldGroupLabelSubDescriptor` instances.



Header Metadata for the Top-Level File Package (partial)
File Body (partial)
Wave Audio Essence Descriptor
ChannelAssignment = UL of AudioLabelingFrameworkADMContent label
SubDescriptors
Sound Track
PCM audio (three channels)
RIFFChunkDefinitionSubDescriptor
RIFFChunkStreamID: UInt32
RIFFChunkID: ISO7 = "axml"
RIFFChunkUUID: UUID [0..1]
RIFFChunkHashSHA1: DataValue [0..1]
Generic Stream Payload
= body of <axml> BW64 RIFF Chunk
(an XML doc containing ADM metadata)
ADMAudioMetadataSubDescriptor
RIFFChunkStreamID_link1: UInt32
ADMProfileLevelULBatch: AUIDSet [0..1]
ADMSoundfieldGroupLabelSubDescriptor
MCALinkID
MCALabelDictionaryID = UL of ADMSoundfield label
MCATagSymbol = "ADM"
MCATagName = "ADM"
RIFFChunkStreamID_link2: UInt32
ADMAudioProgrammeID_ST2131: UTF16String [0..1] = "APR_1001"
ADMAudioContentID_ST2131: UTF16String [0..1]
ADMAudioObjectID_ST2131: UTF16String [0..1]
ADMSoundfieldGroupLabelSubDescriptor
MCALinkID
MCALabelDictionaryID = UL of ADMSoundfield label
MCATagSymbol = "ADM"
MCATagName = "ADM"
RIFFChunkStreamID_link2: UInt32
ADMAudioProgrammeID_ST2131: UTF16String [0..1] = "APR_1002"
ADMAudioContentID_ST2131: UTF16String [0..1]
ADMAudioObjectID_ST2131: UTF16String [0..1]
RIFFChunkReferencesSubDescriptor
RIFFChunkStreamIDsArray: UInt32Array
Ownership
Associations
ADM_CHNASubDescriptor
NumLocalChannels: UInt16 = 3
NumADMAudioTrackUIDs: UInt16 = 4
ADMChannelMappingsArray: ADMChannelMappingStrongReferenceVector
ADMChannelMapping
LocalChannelID: UInt32 = 1
ADMAudioTrackUID: UTF16String
ADMAudioTrackChannelFormatID: UTF16String
ADMAudioPackFormatID: UTF16String [0..1]
ADMChannelMapping
LocalChannelID: UInt32 = 2
ADMAudioTrackUID: UTF16String
ADMAudioTrackChannelFormatID: UTF16String
ADMAudioPackFormatID: UTF16String [0..1]
ADMChannelMapping
LocalChannelID: UInt32 = 3
ADMAudioTrackUID: UTF16String
ADMAudioTrackChannelFormatID: UTF16String
ADMAudioPackFormatID: UTF16String [0..1]
ADMChannelMapping
LocalChannelID: UInt32 = 2
ADMAudioTrackUID: UTF16String
ADMAudioTrackChannelFormatID: UTF16String
ADMAudioPackFormatID: UTF16String [0..1]

Figure 1 – Example: single Essence Track MXF file with all mechanisms defined in this document

C.2 Splitting ADM-described content across Sound Tracks

This example scenario is illustrated in Figure 2 and demonstrated by elements “a” (BW64 PCM+ADM file) and “b” (MXF file) of this document (see Annex D).

This scenario is the same as in C.1 except that:

- The ADM-described content has eight channels of audio. This content is all channel-based rather than object-based, with six channels for some 5.1 content and two channels for some stereo content.
- Two MXF Sound Tracks are created, with the audio from the BW64 file split between them: one Sound Track for the 5.1 content and one Sound Track for the stereo content
- The Audio Labeling Framework for ADM-described Content is not utilized

As a result of using two MXF Sound Tracks, the Multiple Descriptor is employed. Therefore, the `RIFFChunkDefinitionSubDescriptor` instance, and the `ADMAudioMetadataSubDescriptor` instance that accompanies it, are positioned in the Multiple Descriptor rather than in the Wave Audio Essence Descriptors. The ADM metadata in the <axml> BW64 RIFF Chunk is applicable to both Sound Tracks and so is referenced from both of them with `RIFFChunkReferencesSubDescriptor` instances.

The Audio Labeling Framework for ADM-described Content could be utilized and `ADMSoundfieldGroupLabelSubDescriptor` instances added. The instances to create, and where best to position them, would depend on the exact nature of the ADM-described content (as well as any application-specific constraints). Consider these example cases:

- Case 1: The stereo content is dialog and the 5.1 content is some combination of background noise, sound effects and music. The ADM metadata defines several audioProgrammes that combine these in different ways. Each audioProgramme is a finished program mix. One `ADMSoundfieldGroupLabelSubDescriptor` instance could be created for each audioProgramme, with all these instances positioned in the Multiple Descriptor (because each one uses audio from both Sound Tracks).
- Case 2: The stereo content and the 5.1 content are alternative finished program mixes. The ADM metadata defines two audioProgrammes: one for each. One `ADMSoundfieldGroupLabelSubDescriptor` instance could be created for each audioProgramme, with each instance positioned in the relevant Wave Audio Essence Descriptor (because each one uses audio from only one Sound Track). Alternatively, both instances could be positioned in the Multiple Descriptor, if preferred.

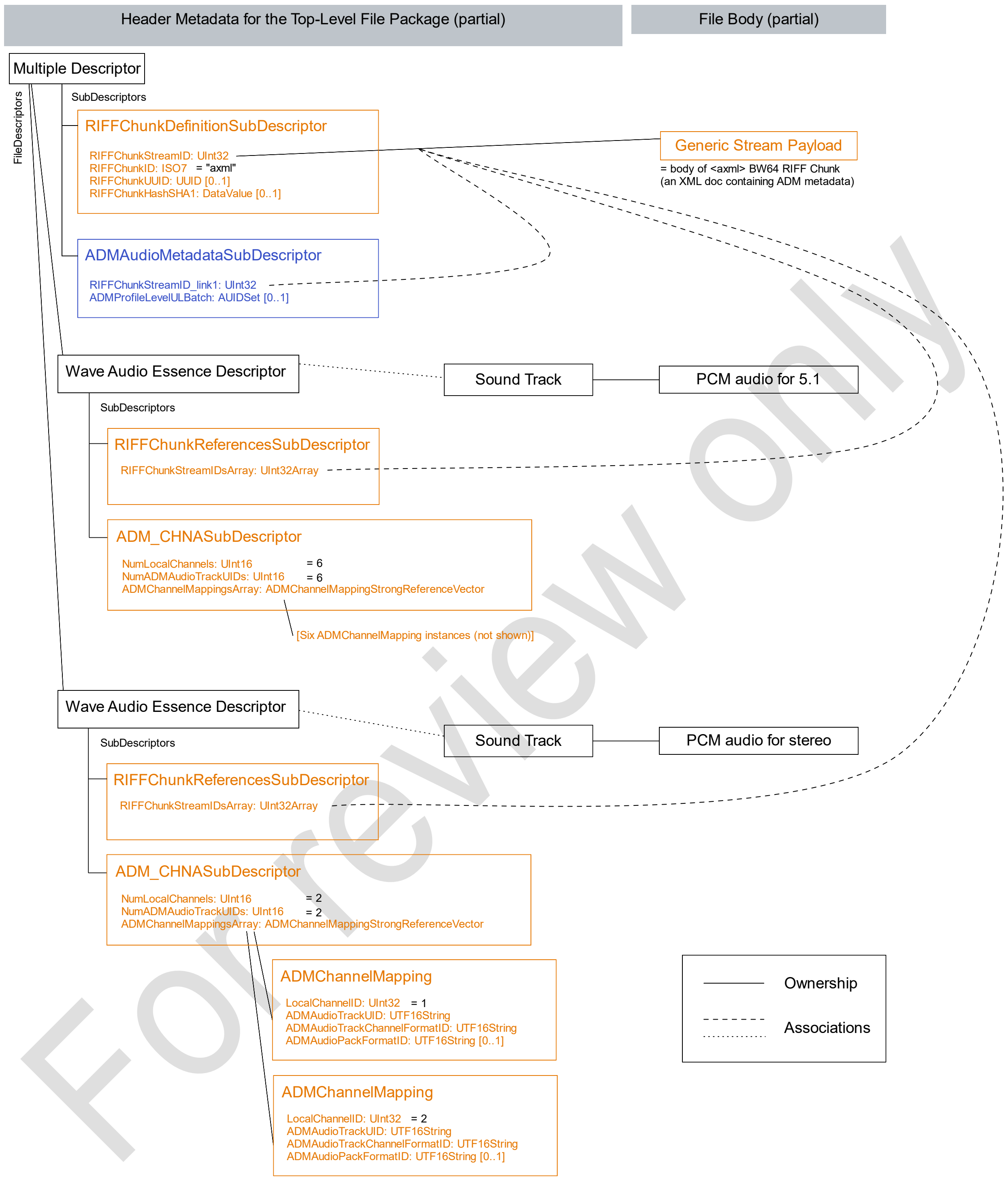

Header Metadata for the Top-Level File Package (partial)
File Body (partial)
Multiple Descriptor
FileDescriptors
SubDescriptors
RIFFChunkDefinitionSubDescriptor
RIFFChunkStreamID: UInt32
RIFFChunkID: ISO7 = "axml"
RIFFChunkUUID: UUID [0..1]
RIFFChunkHashSHA1: DataValue [0..1]
Generic Stream Payload
= body of <axml> BW64 RIFF Chunk
(an XML doc containing ADM metadata)
ADMAudioMetadataSubDescriptor
RIFFChunkStreamID_link1: UInt32
ADMProfileLevelULBatch: AUIDSet [0..1]
Wave Audio Essence Descriptor
Sound Track
PCM audio for 5.1
SubDescriptors
RIFFChunkReferencesSubDescriptor
RIFFChunkStreamIDsArray: UInt32Array
ADM_CHNASubDescriptor
NumLocalChannels: UInt16 = 6
NumADMAudioTrackUIDs: UInt16 = 6
ADMChannelMappingsArray: ADMChannelMappingStrongReferenceVector
[Six ADMChannelMapping instances (not shown)]
Wave Audio Essence Descriptor
Sound Track
PCM audio for stereo
SubDescriptors
RIFFChunkReferencesSubDescriptor
RIFFChunkStreamIDsArray: UInt32Array
ADM_CHNASubDescriptor
NumLocalChannels: UInt16 = 2
NumADMAudioTrackUIDs: UInt16 = 2
ADMChannelMappingsArray: ADMChannelMappingStrongReferenceVector
ADMChannelMapping
LocalChannelID: UInt32 = 1
ADMAudioTrackUID: UTF16String
ADMAudioTrackChannelFormatID: UTF16String
ADMAudioPackFormatID: UTF16String [0..1]
ADMChannelMapping
LocalChannelID: UInt32 = 2
ADMAudioTrackUID: UTF16String
ADMAudioTrackChannelFormatID: UTF16String
ADMAudioPackFormatID: UTF16String [0..1]
Ownership
Associations

Figure 2 – Example: mapping of ADM-described content to an MXF file with two Sound Tracks

C.3 ADM_CHNASubDescriptor only

In this scenario the MXF file created uses only one aspect of SMPTE ST 2131: it contains only `ADM_CHNASubDescriptor` instance(s). There are no RIFF Chunks mapped to the MXF file and so none of the other Header Metadata defined by this document is applicable.

This could be relevant in these example cases:

- Case 1: An MXF file is being created from a BW64 file that contains a <chna> BW64 RIFF Chunk but no RIFF Chunks containing ADM metadata
- Case 2: An `ADM_CHNASubDescriptor` instance is sufficient to describe the audio content because all its ADM IDs are "common definitions" defined in Recommendation ITU-R BS.2094
- Case 3: The associated ADM metadata is stored in the MXF file using some other means, or stored externally

Annex D (Informative) Additional elements

This annex lists non-prose elements of this document.

Non-prose element	File name	Description
a	st2131a-2023.wav	Example BW64 PCM+ADM file containing stereo and 5.1 versions of an audio work. On each channel the corresponding speakerLabel (as given in Recommendation ITU-R BS.2094) is spoken. (Informative)
b	st2131b-2023.mxf	Example frame-wrapped MXF file created from: st2131a-2023.wav using SMPTE ST 382 and SMPTE ST 2131, as explained in clause C.2 (Informative)

Bibliography (Informative)

SMPTE ST 377-41, MXF Multichannel Audio Controlled Vocabulary

SMPTE ST 379-2:2010, Material Exchange Format (MXF) — MXF Constrained Generic Container

SMPTE ST 382, Material Exchange Format — Mapping AES3 and Broadcast Wave Audio into the MXF Generic Container

SMPTE OV 2067-0, Interoperable Master Format

SMPTE ST 2067-2:2020, Interoperable Master Format — Core Constraints

SMPTE ST 2067-8, Interoperable Master Format — Common Audio Labels

SMPTE ST 2127-1, Mapping Metadata-Guided Audio (MGA) signals into the MXF Constrained Generic Container

SMPTE ST 2127-10, Mapping Metadata-Guided Audio (MGA) signals with S-ADM Metadata into the MXF Constrained Generic Container

Recommendation ITU-R BS.2094, Common definitions for the audio definition model

Recommendation ITU-R BS.2125, A serial representation of the Audio Definition Model

Report ITU-R BS.2388, Usage Guidelines for the Audio Definition Model and Multichannel Audio Files

IETF RFC 4122, A Universally Unique IDentifier (UUID) URN Namespace

EBU Tech 3293, EBUCore Metadata Set

EBU Tech 3306, MBWF / RF64: An Extended File Format for Audio