EBU

OPERATING EUROVISION AND EURORADIO

TECH 3364

AUDIO DEFINITION MODEL

METADATA SPECIFICATION

Status: Version 1.0

Geneva
January 2014

Page intentionally left blank. This document is paginated for two sided printing

Conformance Notation

This document contains both normative text and informative text.

All text is normative except for that in the Introduction, any section explicitly labelled as 'Informative' or individual paragraphs which start with 'Note:'.

Normative text describes indispensable or mandatory elements. It contains the conformance keywords 'shall', 'should' or 'may', defined as follows:

'Shall' and 'shall not':	Indicate requirements to be followed strictly and from which no deviation is permitted in order to conform to the document.
'Should' and 'should not':	Indicate that, among several possibilities, one is recommended as particularly suitable, without mentioning or excluding others. OR indicate that a certain course of action is preferred but not necessarily required. OR indicate that (in the negative form) a certain possibility or course of action is deprecated but not prohibited.
'May' and 'need not':	Indicate a course of action permissible within the limits of the document.

Default identifies mandatory (in phrases containing "shall") or recommended (in phrases containing "should") presets that can, optionally, be overwritten by user action or supplemented with other options in advanced applications. Mandatory defaults must be supported. The support of recommended defaults is preferred, but not necessarily required.

Informative text is potentially helpful to the user, but it is not indispensable and it does not affect the normative text. Informative text does not contain any conformance keywords.

A conformant implementation is one which 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.

Page intentionally left blank. This document is paginated for two sided printing

Contents

Audio Definition Model

EBU Committee	*First Issued*	*Revised*	*Re-issued*
TC	2014		

Keywords: BWF, Broadcast Wave Format, File, Audio Definition Model, ADM, Metadata.

1. Introduction

Audio for broadcasting and cinema is evolving towards an immersive and interactive experience which requires the use of more flexible audio formats. A fixed channel-based approach is not sufficient to encompass these developments and so combinations of channel, object and scene-based formats are being developed. ITU-R Reports BS.2266 [1] and BS.1909 [2] highlight these developments and the need for the production chain to accommodate them.

The central requirement for allowing all the different types of audio to be distributed, whether by file or by streaming, is that whatever file/stream format (e.g. Broadcast Wave Format (BWF) [3]) is used, metadata should co-exist to fully describe the audio. Each individual track within a file or stream should be able to be correctly rendered, processed or distributed according to the accompanying metadata. To ensure compatibility across all systems, the Audio Definition Model is an open standard that will make this possible.

2. Background

This purpose of this model is to formalise the description of audio. It is not a format for carrying audio. This distinction will help in the understanding of the model.

2.1 Cooking Analogy

To help explain what the audio definition model (ADM) actually does, it may be useful to consider a cooking analogy. The recipe for a cake will contain a list of ingredients, instructions on how to combine those ingredients and how to bake the cake.

The ADM is like a set of rules for writing the list of ingredients; it gives a clear description of each item, for example: 2 eggs, 400g flour, 200g butter, 200g sugar.

The ADM does not provide the instructions for combining and cooking the ingredients; in the audio world that is what the renderer does.

The BWF <*chna*> chunk is like the bar code on the packet of each of the ingredients; this code allows us to look up the model's description of each item. The bag containing the actual ingredients is like the 'data' chunk of the BWF file that contains the audio samples.

From a BWF file point of view, we would look at our bar codes on each ingredient in our bag, and use that to look up the description of each item in the bag. Each description follows the structure of the model. There might be ingredients such as breadcrumbs which could be divided into its own components (flour, yeast, etc.); which is like having an audio object containing multiple channels (e.g. 'stereo' containing 'left' and 'right').

2.2 Brief Overview

This model will initially use XML as its specification language (with the EBU Core [4] as the target schema), though it could be mapped to other languages such as JSON (JavaScript Object Notation) if required. When it is used with BWF files, the XML can be embedded in the <*axml*> chunk of the file.

The model is divided into two sections, the **content** part, and the **format** part. The content part describes what is contained in the audio, so will describe things like the language of any dialogue, the loudness and so on. The format part describes the technical nature of the audio so it can be decoded or rendered correctly. Some of the format elements may be defined before we have any audio signals, whereas the content parts can usually only be completed after the signals have been generated.

While this model is based around BWF, it is not intended to be solely used for BWF files, but as a more general model. However, using BWF explains more clearly how the model works. It is also expected that the model's parameters are added to in subsequent versions of this specification to reflect the progress in audio technology.

3. Description of the Model

The overall diagram of the model is given in Figure 1. This shows how the elements relate to each other and illustrates the split between the content and format parts. It also shows the <*chna*> chunk of the BWF file and how it connects the tracks in the file to the model.

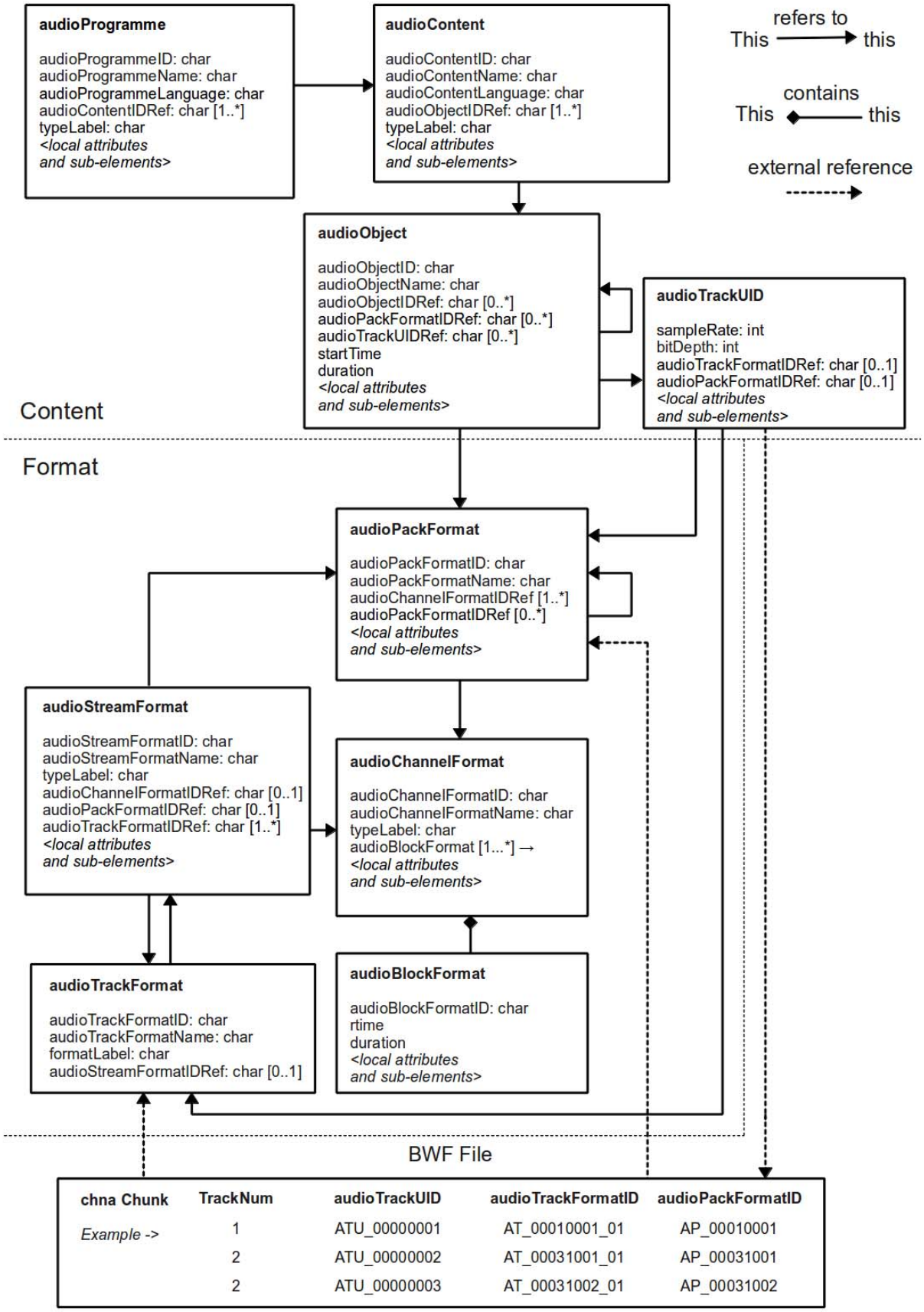


Figure 1: Overall UML Model

Where a BWF file contains a number of audio tracks, it is necessary to know what each track is. The <*chna*> chunk contains a list of numbers corresponding to each track in the file. Hence, for a 6 track file, the list is at least 6 long. For each track there is an audioTrackFormatID number and an audioTrackUID number (notice the additional 'U' which stands for 'unique'). The reason the list could be longer than the number of tracks is that a single track may have different definitions at different times so will require multiple audioTrackUIDs and references.

The audioTrackFormatID is used to look up the definition of the format of that particular track. The audioTrackFormatIDs are not unique; for example, if a file contains 5 stereo pairs, there will be 5 identical audioTrackFormatIDs to describe the 'left' channel, and 5 to describe the 'right' channel. Thus, only two different audioTrackFormatIDs will need to be defined. However, audioTrackUIDs are unique (hence the 'U'), and they are there to uniquely identify the track. This use of IDs means that the tracks can be ordered in any way in the file; their IDs reveal what those tracks are.

3.1 Format

The audioTrackFormatID in the <*chna*> chunk answers the question "What is the format of this track?" The audioTrackFormatID will also contain an audioStreamFormatID, which allows identification of the combination of the audioTrackFormat and audioStreamFormat. An audioStreamFormat describes a decodable signal - PCM signal or a Dolby E stream for example.

The audioStreamFormat is made up of one or more audioTrackFormats. For example, the audioStreamFormat of a PCM signal will contain only one audioTrackFormat, but a Dolby E audioStreamFormat will contain two audioTrackFormats. In other words two tracks have to be combined to decode a Dolby E stream. Hence, the combination of audioStreamFormat and audioTrackFormat reveals whether the signal extracted from the BWF may be used directly (if it is PCM), or if it has to be decoded (if it is Dolby E).

The next stage is to find out what type of audio the stream is; for example it may be a conventional channel (e.g. 'front left'), an audio object (e.g. something named 'guitar' positioned at the front), a HOA (Higher Order Ambisonics) component (e.g. 'X') or a group of channels (e.g. a Dolby E stream containing 5.1). Inside audioStreamFormat there will be a reference to either an audioChannelFormat or audioPackFormat that will describe the audio stream. There will only be one of these references.

If audioStreamFormat contains an audioChannelFormat reference (i.e. audioChannelFormatIDRef) then audioStreamFormat is one of several different types of audioChannelFormat. An audioChannelFormat is a description of a single waveform of audio. In audioChannelFormat there is a typeDefinition attribute, which is used to define what the type of channel is.

The typeDefinition attribute can be set to 'DirectSpeakers', 'HOA', 'Matrix' 'Objects' or 'Binaural'. For each of those types, there is a different set of sub-elements to specify the static parameters associated with that type of audioChannelFormat. For example, the 'DirectSpeakers' type of channel has the sub-element 'speakerLabel' for allocating a loudspeaker to the channel.

To allow audioChannelFormat to describe dynamic channels (i.e. channels that change in some way over time), it uses audioBlockFormat to divide the channel along the time axis. The audioBlockFormat element will contain a start time (relative to the start time of the parent audioObject) and duration. Within audioBlockFormat there are time-dependent parameters that describe the channel which depend upon the audioChannelFormat type.

For example, the 'Objects' type of channel has the sub-elements 'azimuth', 'elevation' and 'distance' to describe the location of the sound. The number and duration of audioBlockFormats is not limited, there could be an audioBlockFormat for every sample if something moves rapidly, though that might be a bit excessive! At least one audioBlockFormat is required and so static channels will have one audioBlockFormat containing the channel's parameters.

If audioStreamFormat refers to an audioPackFormat, it describes a group of channels. An audioPackFormat element groups together one or more audioChannelFormats that belong together (e.g. a stereo pair). This is important when rendering the audio, as channels within the group may need to interact with each other.

The reference to an audioPackFormat containing multiple audioChannelFormats from an audioStreamFormat usually occurs when the audioStreamFormat contains non-PCM audio such as Dolby E which carries several channels encoded together. AudioPackFormat would usually not be referred from audioStreamFormat for most channel and scene-based formats with PCM audio. Where this reference does exist, the function of audioPackFormat is to combine audioChannelFormats that belong together for rendering purposes.

For example, 'stereo', '5.1', '1st order ambisonics' would all be examples of an audioPackFormat. Note that audioPackFormat just describes the format of the audio. For example, a file containing 5 stereo pairs will contain only one audioPackFormat to describe 'stereo'. It is possible to nest audioPackFormats; a '2nd order HOA' could contain a '1st order HOA' audioPackFormat alongside audioChannelFormats for the R, S, T, U & V components.

3.2 Content

Using the 5 stereo pair file as an example, the audioTrackFormat defines which tracks are left and right, not which ones belong together, nor what is represented in them. AudioObject is used to determine which tracks belong together and where they are in the file. This element links the actual audio data with the format, and this is where audioTrackUID comes in. For a stereo pair (in PCM), audioObject will contain references to two audioTrackUIDs; therefore, those two tracks will contain stereo audio. It will also contain a reference to audioPackFormat, which defines the format of those two tracks as a stereo pair.

As there are 5 stereo pairs in this example, 5 audioObject elements will be needed. Each one will contain the same reference to a stereo audioPackFormat, but will contain different reference to audioTrackUIDs, as each stereo pair is carrying different audio. The order of audioTrackUIDRefs is not important in an audioObject, as the format definition through audioTrack, audioStreamFormat, audioChannelFormat and audioPackFormat determines which track is which.

The audioObject element also contains startTime and duration attributes. This start time is the time when the signal for the object starts in the file. Thus, if startTime is "00:00:10.00000", the signal for the object will start 10 seconds into the track in the BWF file.

As audioPackFormat can be nested, it follows that audioObjects can be nested. For example if a Dolby E pair of tracks contains 5.1+2.0, there are two groups which need to be treated differently. Therefore, the audioObject will contain not only references to the two audioTrackUIDs carrying the stream, but also references to two audioObjects, one for the 5.1 and one for the 2.0.

AudioObject is referred to by audioContent, which gives a description of the content of the audio; it has parameters such as language (if there's dialogue) and the R128 loudness parameters. Some of the values for these parameters can only be calculated after the audio has been generated, and this is why they are not in the format part.

AudioProgramme brings all the audioContent together; it combines them to make the complete 'mix'.

For example:

- an audioProgramme may contain audioContent for 'narrator' and another one for 'background music'
- an audioProgramme for France may contain audioContents called ‘dialogue-fr’ and ‘backgroundMusic’, and another audioProgramme for the UK which contains audioContents called ‘dialogue-en’ and the same ‘backgroundMusic’.

4. Standard Formats

For many situations, particularly in channel and scene-based work, many of the required formats will be common. For example, mono, stereo and 5.1 all have standard definitions and it would be inefficient to generate and carry a mass of XML every time one of these formats needs to be described. Therefore, the EBU plans to generate a set of standard format descriptions for many of the commonly used formats.

This set will be freely available in a reference XML file, which will be updated regularly. The reference file will not have to be included in the BWF <*axml*> chunk, and can be externally referred to. Therefore a BWF file will not need to carry the XML of the format if only standard formats are used. The occasions any ADM XML code will need to be carried in the <*axml*> chunk is when audioProgramme, audioContent and audioObject are used, or custom definitions are required.

5. ADM Elements

Each of the elements within the ADM is described in the following subsections. The attributes and sub-elements marked with an asterisk (*) are already defined in the EBU Core Metadata set [4].

5.1 audioTrackFormat

The audioTrackFormat element corresponds to a single set of samples or data in a single track in a storage medium. It is used to describe what format the data is in, allowing a renderer to decode the signal correctly. It is referred from the audioStreamFormat element which is used to identify the combination of tracks required to decode the track data successfully.

For PCM audio an audioStreamFormat will refer to a single audioTrackFormat and so the two elements are effectively describing the same thing. For coded audio such as Dolby E, multiple audioTrackFormats will have to be combined in a single audioStreamFormat to generate decodable data.

Software that parses the model can start from either audioTrackFormat or audioStreamFormat. To allow for this flexibility audioTrackFormat can also refer back to the audioStreamFormat. However, it is a strict requirement that if this reference is used, the audioTrackFormat must refer to the same audioStreamFormat that is referring back to it.

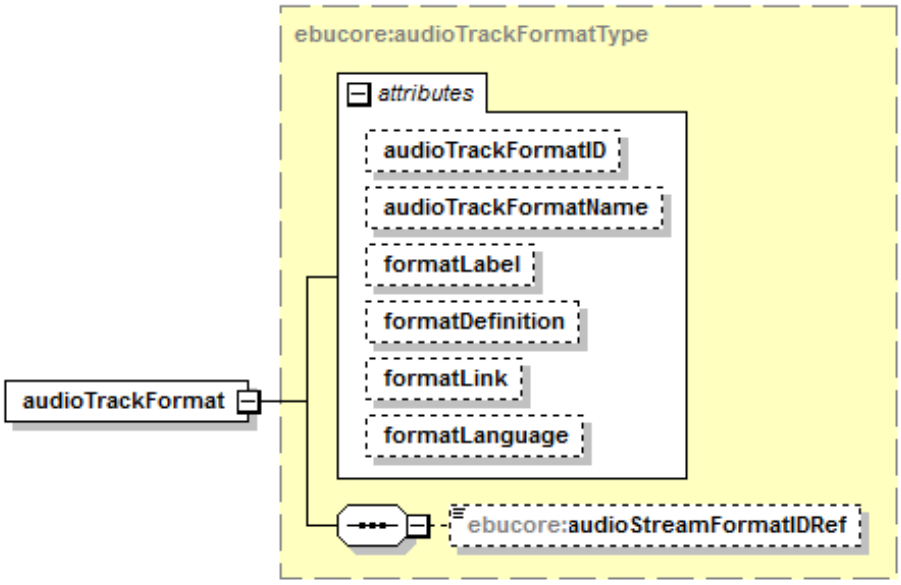


Figure 2: audioTrackFormat

5.1.1 Attributes

Attribute	Description	Example
audioTrackFormatID	ID for track	AT_00010001_01
audioTrackFormatName	Name for track	PCM_FrontLeft
formatLabel*	Descriptor of the format	0001
formatDefinition*	Description of the format	PCM
formatLink*	URI for the format (not currently used in the ADM)	
formatLanguage*	Language of the formatDefinition (not currently used in the ADM)	

5.1.2 Sub-elements

Element	Description	Example	Quantity
audioStreamFormatIDRef	Reference to an audioStreamFormat	AS_00010001	0 or 1

5.1.3 Sample Code

```
<audioTrackFormat audioTrackFormatID="AT_00010001_01" audioTrackFormatName="PCM_FrontLeft"
formatDefinition="PCM" formatLabel="0001">
  <audioStreamFormatIDRef>AS_00010001</audioStreamFormatIDRef>
</audioTrackFormat>
```

5.2 audioStreamFormat

A stream is a combination of tracks (or one track) required to render a channel, object, HOA component or pack. The audioStreamFormat establishes a relationship between audioTrackFormats and the audioChannelFormats or audioPackFormat. Its main use is to deal with non-PCM encoded tracks, where one or more audioTrackFormats must be combined to represent a decodable signal that covers several audioChannelFormats (by referencing an audioPackFormat). For example a Dolby E audioStreamFormat will refer to two audioTrackFormats (for the two tracks carrying the Dolby E bitstream) and an audioPackFormat for the 5.1 channel arrangement it has encoded.

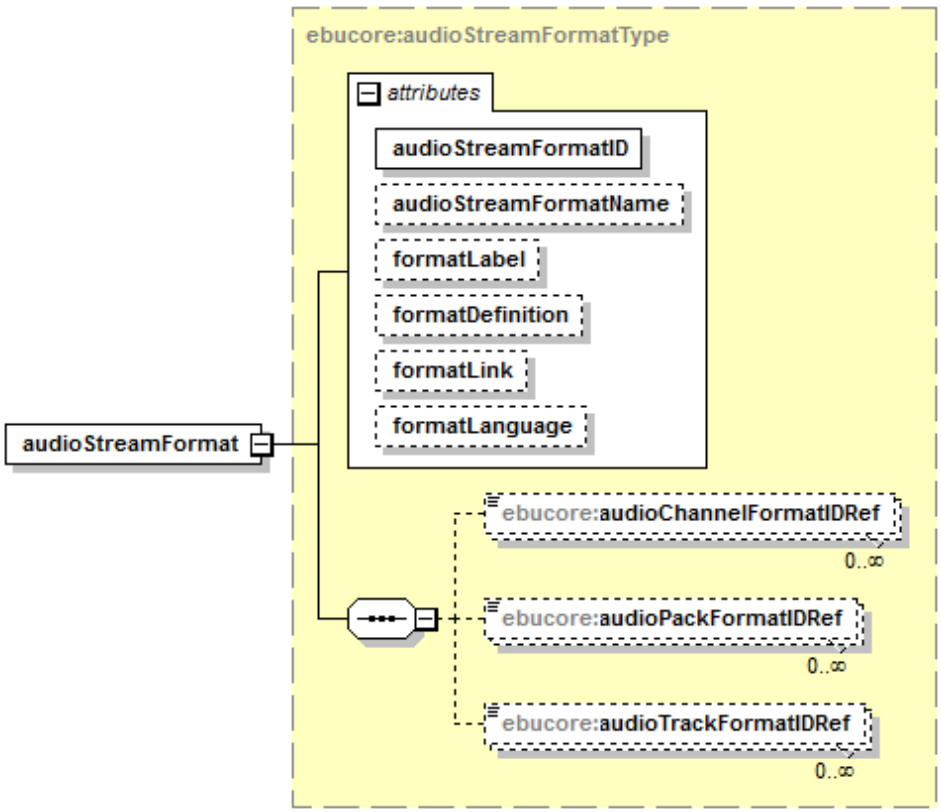


Figure 3: audioStreamFormat

5.2.1 Attributes

Attribute	Description	Example
audioStreamFormatID	ID for the stream	AS_00010001
audioStreamFormatName	Name of the stream	PCM_FrontLeft
formatLabel*	Descriptor of the format	0001
formatDefinition*	Description of the format	PCM
formatLink*	URI for the format (not currently used in the ADM)	
formatLanguage*	Language of format description (not currently used in the ADM)	

5.2.2 Sub-elements

Element	Description	Example
audioChannelFormatIDRef	Reference to audioChannelFormat	AC_00010001
audioPackFormatIDRef	Reference to audioPackFormat	AP_00010003
audioTrackFormatIDRef	Reference to audioTrackFormat	AT_00010001_01

5.2.3 Sample Code

```
<audioStreamFormat audioStreamFormatID="AT_00010001" audioStreamFormatName="PCM_FrontLeft"
formatDefinition="PCM"
formatLabel="0001">
  <audioTrackFormatIDRef>AT_00010001_01</audioTrackFormatIDRef>
  <audioChannelFormatIDRef>AC_00010001</audioChannelFormatIDRef>
</audioStreamFormat>
```

5.3 audioChannelFormat

An audioChannelFormat represents a single sequence of audio samples on which some action may be performed, such as movement of an object which is rendered in a scene. It is sub-divided in the time domain into one or more audioBlockFormats.

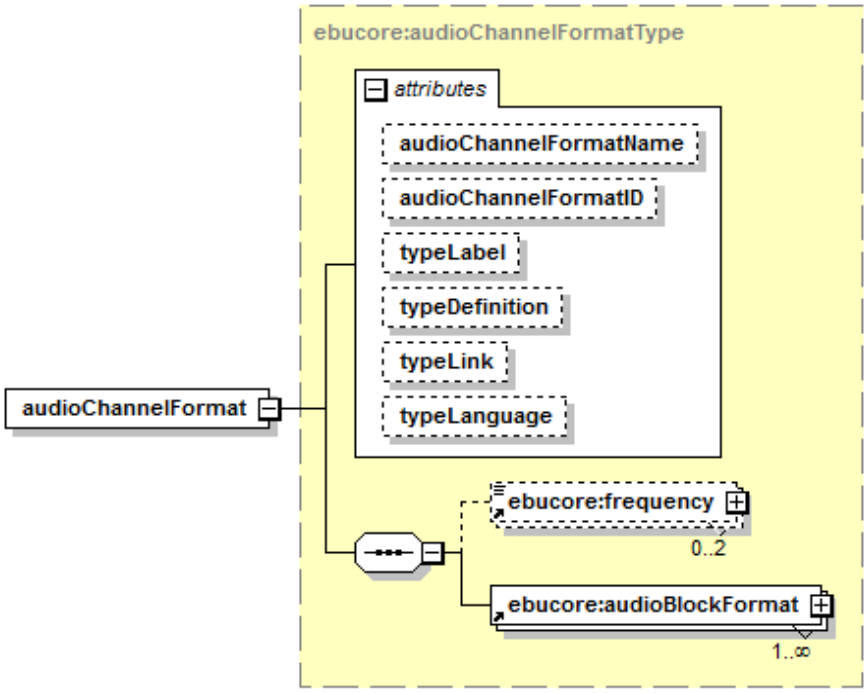


Figure 4: audioChannelFormat

5.3.1 Attributes

Attribute	Description	Example
audioChannelFormatName	Name of the channel	FrontLeft
audioChannelFormatID	ID of the channel	AC_00010001
typeLabel*	Descriptor of the type of channel	0001
typeDefinition*	Description of the type of channel	DirectSpeakers
typeLink*	URI for the type (not currently used in the ADM)	
typeLanguage*	Language of the typeDefinition (not currently used in the ADM)	

The typeDefinition of the audioChannel Format specifies the type of audio it is describing, and also determines which parameters are used within its audioBlockFormat children.

Currently, there are five different typeDefinitions:

typeDefinition	typeLabel	Description
DirectSpeakers	0001	For channel-based audio, where each channel feeds a speaker directly
Matrix	0002	For channel-based audio where channels are matrixed together, such as Mid-Side, Lt/Rt
Objects	0003	For object-based audio where channels represent audio objects (or parts of objects), so include positional information
HOA	0004	For scene-based audio where Ambisonics and HOA are used
Binaural	0005	For binaural audio, where playback is over headphones

5.3.2 Sub-elements

Element	Description	Attributes	Quantity
audioBlockFormat	Time division of channel containing dynamic metadata	see Section 5.4	1…*
frequency	Sets a high or low cut-off frequency for the audio in Hz	typeDefinition = "lowPass" or "highPass"	0…2

The optional frequency parameter allows a frequency range of the audio to be specified. This can be either low-pass or high-pass, or by combining both to achieve band-pass and band-stop. The mostly common use of this is for LFE channels where a low-pass frequency (e.g. 200 Hz) can be specified.

5.3.3 Sample Code

```
<audioChannelFormat audioChannelFormatID="AC_00010001" audioChannelFormatName="FrontLeft"
typeDefinition="DirectSpeakers">
  <audioBlockFormat ...>
    ...
  </audioBlockFormat>
</audioChannelFormat>
```

5.4 audioBlockFormat

An audioBlockFormat represents a single sequence of audioChannelFormat samples with fixed parameters, including position, within a specified time interval.

5.4.1 Attributes

Attribute	Description	Example
audioBlockFormatID	ID for block	AB_00010001_00000001
rtime	Start time of block	00:00:00.00000
duration	Duration of block	00:00:05.00000

The sub-elements within audioBlockFormat are dependent upon the typeDefinition or typeLabel of the parent audioChannelFormat element.

Currently, there are five different typeDefinitions:

typeDefinition	typeLabel	Description
DirectSpeakers	0001	For channel-based audio, where each channel feeds a speaker directly
Matrix	0002	For channel-based audio where channels are matrixed together, such as Mid-Side, Lt/Rt
Objects	0003	For object-based audio where channels represent audio objects (or parts of objects) and so include positional information
HOA	0004	For scene-based audio where Ambisonics and HOA are used
Binaural	0005	For binaural audio, where playback is over headphones

5.4.2 Sample Code

```
<audioBlockFormat audioBlockFormatID="AB_00010001_00000001" rtime="00:00:00.00000"
duration="00:00:05.00000">
  ...
</audioBlockFormat>
```

5.4.3 Sub-elements

5.4.3.1 If audioChannelFormat.typeDefinition == "DirectSpeakers"

For channel-based systems, this is the metadata used to describe the channel. If the channel is intended to be played out through a specific loudspeaker, then use *speakerLabel* to indicate the label of that speaker. While both the maximum and minimum values for the three position elements are available (using the bound attribute), they should be avoided as the exact position should normally be specified by omitting the bound attribute.

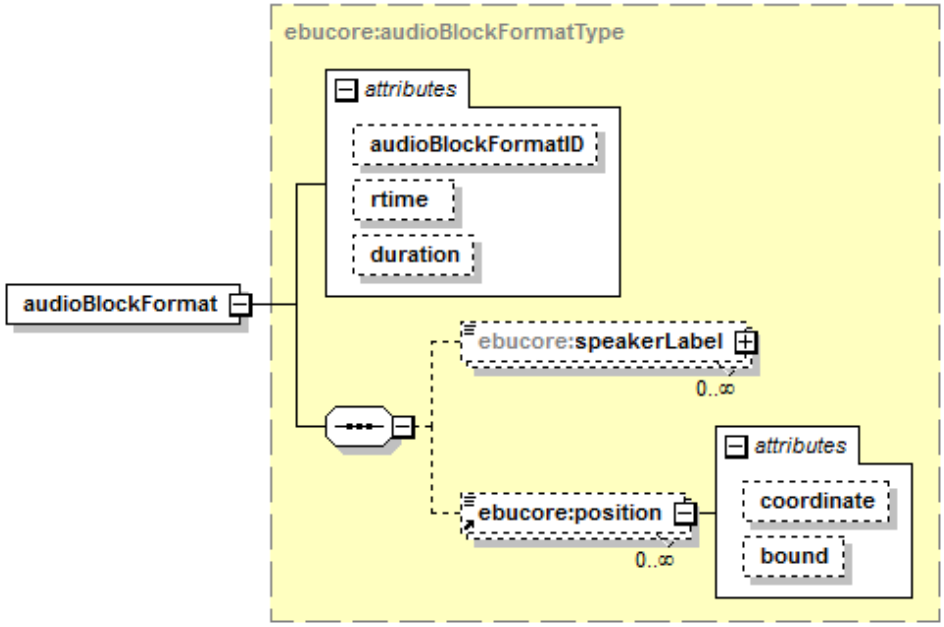


Figure 5: audioBlockFormat (DirectSpeakers)

Element	co-ordinate attribute	bound attribute	Description	Units	Example	Quantity
speakerLabel	N/A	N/A	A reference to the label of the speaker position	-	M-30	0...*
position	azimuth		Exact azimuth location of sound	Degrees	-30.0	1
position	azimuth	max	Max. azimuth location of sound	Degrees	-22.5	0 or 1
position	azimuth	min	Min. azimuth location of sound	Degrees	-30.0	0 or 1
position	elevation		Exact elevation location of sound	Degrees	0.0	1
position	elevation	max	Max. elevation location of sound	Degrees	5.0	0 or 1
position	elevation	min	Min. elevation location of sound	Degrees	0.0	0 or 1
position	distance		Exact normalised distance from origin	Normalised to 1	1.0	0 or 1
position	distance	max	Max. normalised distance from origin	Normalised to 1	0.8	0 or 1
position	distance	min	Min. normalised distance from origin	Normalised to 1	0.9	0 or 1

The distance measure is normalised, as absolute speaker distances from the origin are rarely used, but an absolute reference distance is available in audioPackFormat. These coordinates are based on the polar system, as this is the common way of describing channel and speaker locations. However it is also possible to use the Cartesian coordinate system by using different coordinate attributes ('X', 'Y' and 'Z'); and this system is described in more detail in Section 8.

5.4.3.1.1 Sample Code

```
<audioBlockFormat ...>
  <speakerLabel>M-30</speakerLabel>
  <position coordinate="azimuth">-30.0</position>
  <position coordinate="elevation">0.0</position>
  <position coordinate="distance">1.0</position>
</audioBlockFormat>
```

5.4.3.2 If audioChannelFormat.typeDefinition == "Matrix"

This is for channel-based matrix channels, such as mid-side and Lt/Rt. The matrix element contains a list of coefficient sub-elements which each refer to other channels and a multiplication factor. All the coefficients in this list should be added together to generate the matrix equation.

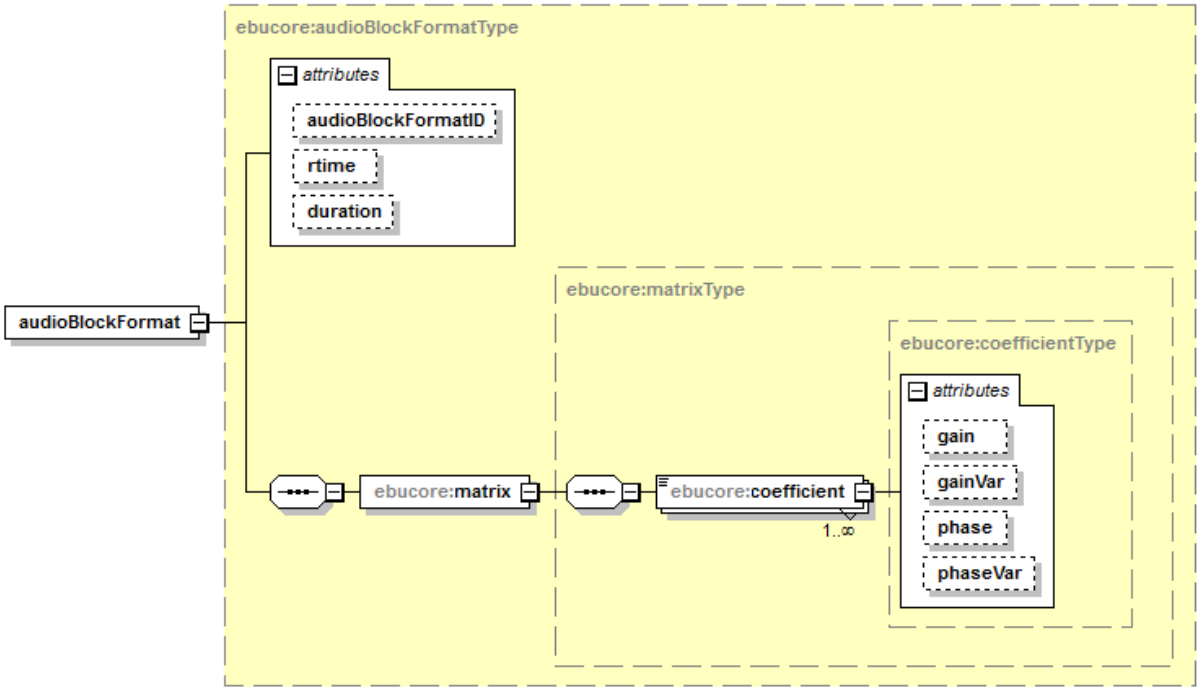


Figure 6: audioBlockFormat (Matrix)

For example, the matrix element of a 'Side' channel will contain two coefficient sub-elements, one with the value 1.0 referring to "Left" and the other with a value of -1.0 referring to 'Right'; this gives Side=Left-Right.

The values for gain and phase shift can either be constants (using gain and phase) or they may be variables (using gainVar and phaseVar) that allow the renderer to decide the value, maybe via another source of metadata. This type of channel can also be used to specify downmixes of channels, such as Lo/Ro.

Element	Sub-elements	Description	Quantity
matrix	coefficient	Contains the coefficients for combining other channels	1

Sub-element	Attribute	Description	Units	Example	Quantity
coefficient	gain	Multiplication factor of another channel. Constant value.	Ratio	-0.5	1...*
coefficient	gainVar	Multiplication factor of another channel. Variable.	Ratio	csel	1...*
coefficient	phase	Phase shift of another channel. Constant value.	Degrees	90	1...*
coefficient	phaseVar	Phase shift of another channel. Variable.	Degrees	ph	1...*
coefficient		Reference to other channel definition		AC_00010001	1...*

5.4.3.2.1 Sample Code

```
<audioBlockFormat ...>
 <matrix>
   <coefficient gain="1.0">AC_00010001</coefficient>
   <coefficient gain="-1.0">AC_00010002</coefficient>
 </matrix>
</audioBlockFormat>
```

5.4.3.3 If audioChannelFormat.typeDefinition == "Objects"

This is for object-based audio where the position of the audio object may change dynamically. As well as the polar coordinates of the object, there are parameters for the object's size, and whether it is a diffuse or coherent sound.

The channelLock parameter will inform a renderer to send the object's audio to the nearest speaker or channel, rather than the usual panning, interpolation, etc. The jumpPosition parameter will ensure the renderer does not perform any temporal interpolation of the position values, so the object will jump in space rather than move smoothly to the next position.

The position elements use the coordinate attribute to specify which axis is used. The primary coordinate system is polar which uses azimuth, elevation and distance axes. However, it is possible to specify other axes for other coordinates such as X, Y and Z for the Cartesian coordinate system. This is described in more detail in Section 8.

Sub-element	attribute	Description	Units	Example	Quantity	Default
position	coordinate= "azimuth"	azimuth of sound location	Degrees	-22.5	1	
position	coordinate= "elevation"	elevation of sound location	Degrees	5.0	1	
position	coordinate= "distance"	distance from origin	Normalised to 1	0.9	0 or 1	1.0
gain		Apply a gain to the audio in the object	linear gain value	0.5	0 or 1	1.0
diffuse		Diffuse or direct sound	1/0 flag	1	0 or 1	0
width		horizontal extent	Degrees	45	0 or 1	0.0
height		vertical extent	Degrees	20	0 or 1	0.0
depth		distance extent	Ratio	0.2	0 or 1	0.0
channelLock		If set to 1 a renderer can lock the object to the nearest channel or speaker, rather than normal rendering.	1/0 flag	1	0 or 1	0
jumpPosition		If set to 1 the position will not be interpolated with the previous block.	1/0 flag	1	0 or 1	0

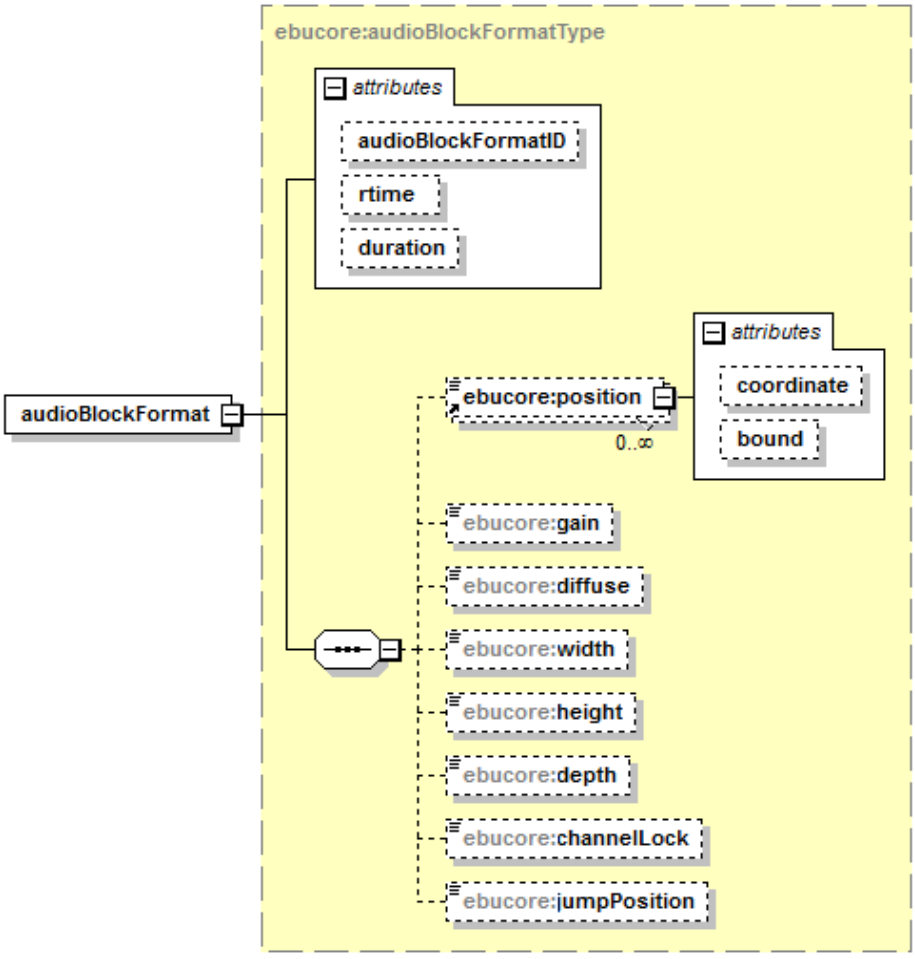


Figure 7: audioBlockFormat (Objects)

5.4.3.3.1 Sample Code

```
<audioBlockFormat ...>
  <position coordinate="azimuth">-22.5</position>
  <position coordinate="elevation">5.0</position>
  <position coordinate="distance">0.9</position>
  <depth>0.2</depth>
</audioBlockFormat>
```

5.4.3.4 If audioChannelFormat.typeDefinition == "HOA"

This is for scene-based channels (or components) such as Ambisonics/HOA. The component can be described by either a combination of degree and order values, or an equation. The different versions of Ambisonics (such as N3D and FuMa) are specified by providing suitable names for the parent audioChannelFormat and audioPackFormat elements. Each version should be assigned a range of ID values to accommodate a sufficient number of channels. It is recommended that C-style mathematical notation be used for the equation element (e.g. 'cos(A)*sin(E)').

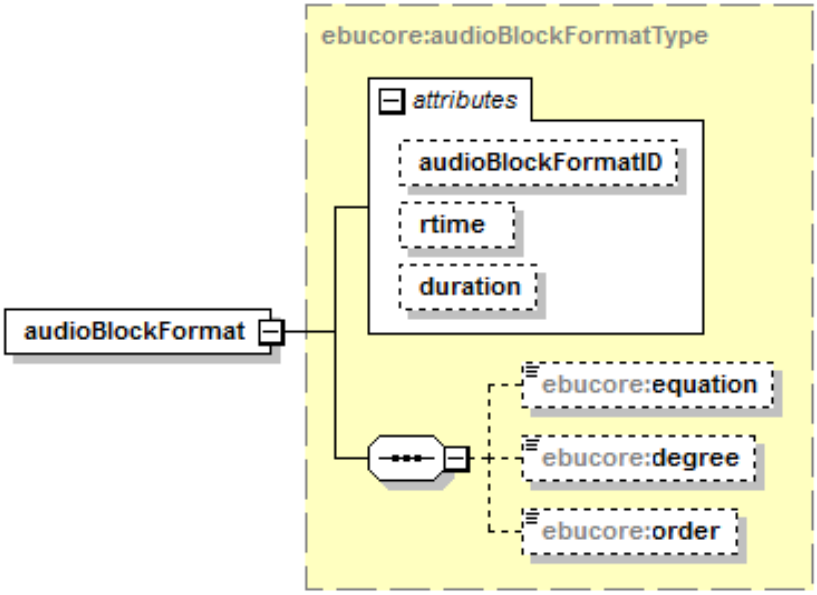


Figure 8: audioBlockFormat (HOA)

Sub-element	Description	Type	Example	Quantity
equation	An equation to describe the HOA component	string		0 or 1
degree	Degree of the HOA component	int	1	0 or 1
order	Order of the HOA component	int	1	0 or 1

5.4.3.4.1 Sample Code

```
<audioBlockFormat ...>
  <degree>1</degree>
  <order>1</order>
</audioBlockFormat>
```

5.4.3.5 If audioChannelFormat.typeDefinition == "Binaural"

This is for binaural representation of audio. Given that binaural consists of two channels, the left and right ear, this is rather simple. As the name of the audioChannelFormat will be either "leftEar" or "rightEar" there is no other metadata required in audioBlockFormat.

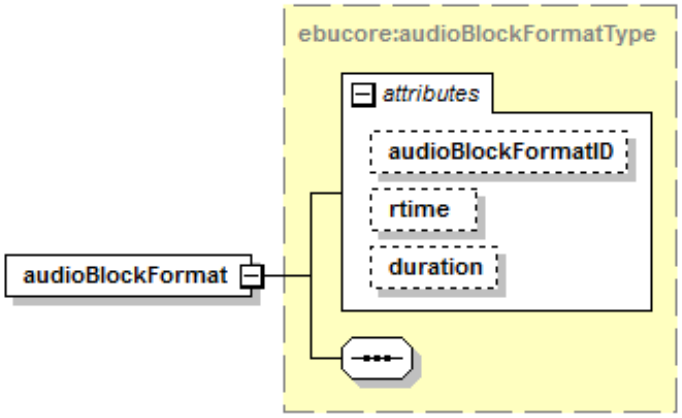


Figure 9: audioBlockFormat (Binaural)

5.4.3.6 Sample Code

```
<audioBlockFormat .../>
```

5.5 audioPackFormat

The audioPackFormat groups together one or more audioChannelFormats that belong together.

Examples of audioPackFormats are 'stereo' and '5.1' for channel-based formats. It can also contain references to other packs to allow nesting. The typeDefinition is used to define the type of channels described within the pack. The typeDefinition/typeLabel must match those in the referred audioChannelFormats.

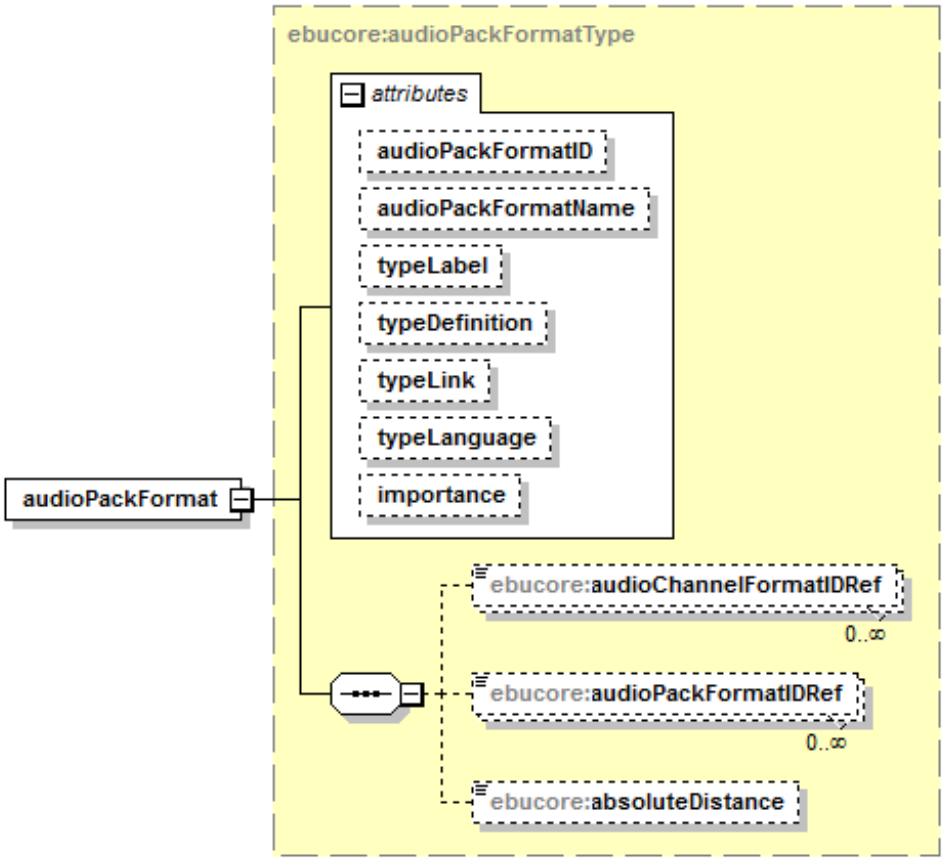


Figure 10: audioPackFormat

5.5.1 Attributes

Attribute	Description	Example
audioPackFormatID	ID for the pack	AP_00010001
audioPackFormatName	Name for the pack	stereo
typeLabel*	Descriptor of the type of channel	0001
typeDefinition*	Description of the type of channel	DirectSpeakers
typeLink*	URI for the type (not currently used in the ADM)	
typeLanguage*	Language of the typeDefinition (not currently used in the ADM)	
importance	Importance of a pack. Allows a renderer to discard a pack below a certain level of importance. 10 is the most important, 0 is the least.	10

There are five different typeDefinitions:

typeDefinition	typeLabel	Description
DirectSpeakers	0001	For channel-based audio, where each channel feeds a speaker directly
Matrix	0002	For channel-based audio where channels are matrixed together, such as Mid-Side, Lt/Rt
Objects	0003	For object-based audio where channels represent audio objects (or parts of objects), so include positional information
HOA	0004	For scene-based audio where Ambisonics and HOA are used
Binaural	0005	For binaural audio, where playback is over headphones

5.5.2 Sub-elements

Element	Description	Example	Quantity
audioChannelFormatIDRef	Reference to an audioChannelFormat	AC_00010001	0…*
audioPackFormatIDRef	Reference to an audioPackFormat	AP_00010002	0…*
absoluteDistance	Absolute distance in metres	4.5	0 or 1

There is an overall absolute distance parameter which can be used with the normalised distance parameters specified with the audioBlockFormats, to give absolute distances to each block.

5.5.3 Sample Code

```
<audioPackFormat audioPackFormatID="AP_000010002" audioPackFormatName="stereo"
typeLabel="0001">
  <audioChannelIDRef>AC_00010001</audioChannelIDRef>
  <audioChannelIDRef>AC_00010002</audioChannelIDRef>
</audioBlockFormat>
```

5.6 audioObject

An audioObject establishes the relationship between the content, the format via audio packs, and the assets using the track UIDs. AudioObjects can be nested and so they can refer to other audioObjects.

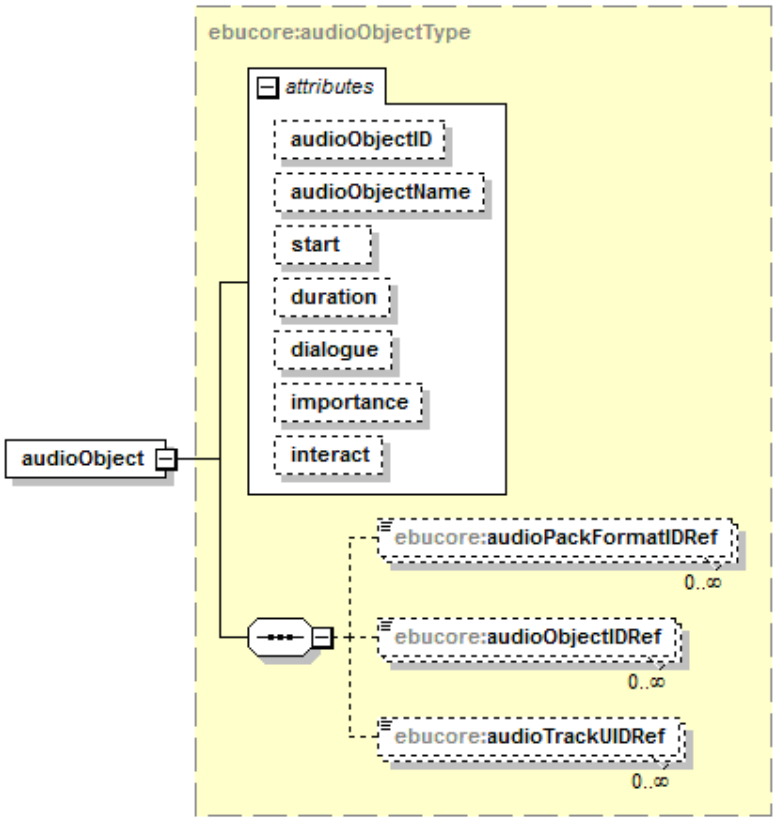


Figure 11: audioObject

5.6.1 Attributes

Attribute	Description	Example
audioObjectID	ID of the object	AO_1001
audioObjectName	Name of the object	dialogue_stereo
Start	Start time for the object, relative to the start of the programme	00:00:00.00000
Duration	Duration of object	00:02:00.00000
Dialogue	If the audio is not dialogue set a value of 0; if it contains only dialogue a value of 1; if it contains both then a value of 2.	0
Importance	Importance of an object. Allows a renderer to discard an object below a certain level of importance. 10 is most important, 0 least.	10
Interact	Set to 1 if a user can interact with the object, 0 if not.	1

5.6.2 Sub-elements

Element	Description	Example
audioPackIDRef	Reference to an audioPack for format description	AP_00010001
audioObjectIDRef	Reference to another audioObject	AO_1002
audioTrackUIDRef	Reference to an audioTrackUID (when using a BWF file this is listed in the <*chna*> chunk)	ATU_00000001

5.6.3 Sample Code

```
<audioObject audioObjectID="AO_1001" audioObjectName="Dialogue_stereo">
  <audioPackIDRef>AP_00010001</audioPackIDRef>
  <audioTrackUIDRef>ATU_00000001</audioTrackUIDRef>
  <audioTrackUIDRef>ATU_00000002</audioTrackUIDRef>
</audioObject>
```

5.7 audioContent

An audioContent element describes the content of one component of a programme (e.g. background music), and refers to audioObjects to tie the content to its format. This element includes loudness metadata which allows EBU R 128 [5] (or other methods) values to be included.

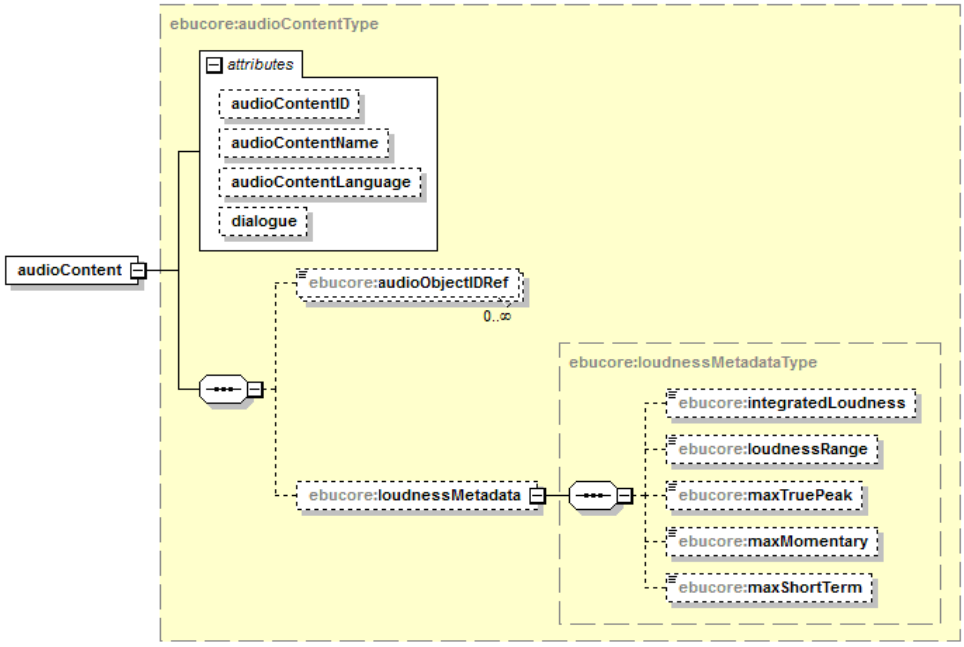


Figure 12: audioContent

5.7.1 Attributes

Attribute	Description	Example
audioContentID	ID of the content	ACO_1001
audioContentName	Name of the content	Music
audioContentLanguage	Language of the content	en
Dialogue	If the audio is not dialogue set a value of 0; if it contains only dialogue set a value of 1; if it contains both then set a value of 2.	0

5.7.2 Sub-elements

Element	Description	Example
audioObjectIDRef	Reference to audioObject	AO_1001
loudnessMetadata	See Section 5.7.3	

5.7.3 Loudness attributes and sub-elements

Attribute	Description	Example
loudnessMethod	The method or standard used to calculate the loudness metadata	R 128

Element	Description	Units	Example
integratedLoudness	Integrated loudness value	LUFS	-23.0
loudnessRange	Loudness range	LU	10.0
maxTruePeak	Maximum true-peak	dBTP	-2.3
maxMomentary	Maximum momentary loudness	LUFS	-19.0
maxShortTerm	Maximum short term loudness	LUFS	-21.2

Note: For historical reasons, some documents use LKFS as loudness units. Measurements in LUFS and LKFS are identical, but the use of LUFS is in accordance with international standards for the naming of units, whilst that of LKFS is not.

5.7.4 Sample Code

```
<audioContent audioContentID="ACO_1001" audioContentName="Music">
  <audioObjectIDRef>AO_1001</audioObjectIDRef>
  <loudnessMetadata>
    <integratedLoudness>-23.0</integratedLoudness>
    <maxTruePeak>-2.3</maxTruePeak>
  </loudnessMetadata>
</audioContent>
```

5.8 audioProgramme

An audioProgramme element refers to a set of one or more audioContents that are combined to create a full audio programme. It contains start and end timecodes for the programme which can be used for alignment with video timecodes. Loudness metadata is also included to allow the programme's loudness to be recorded.

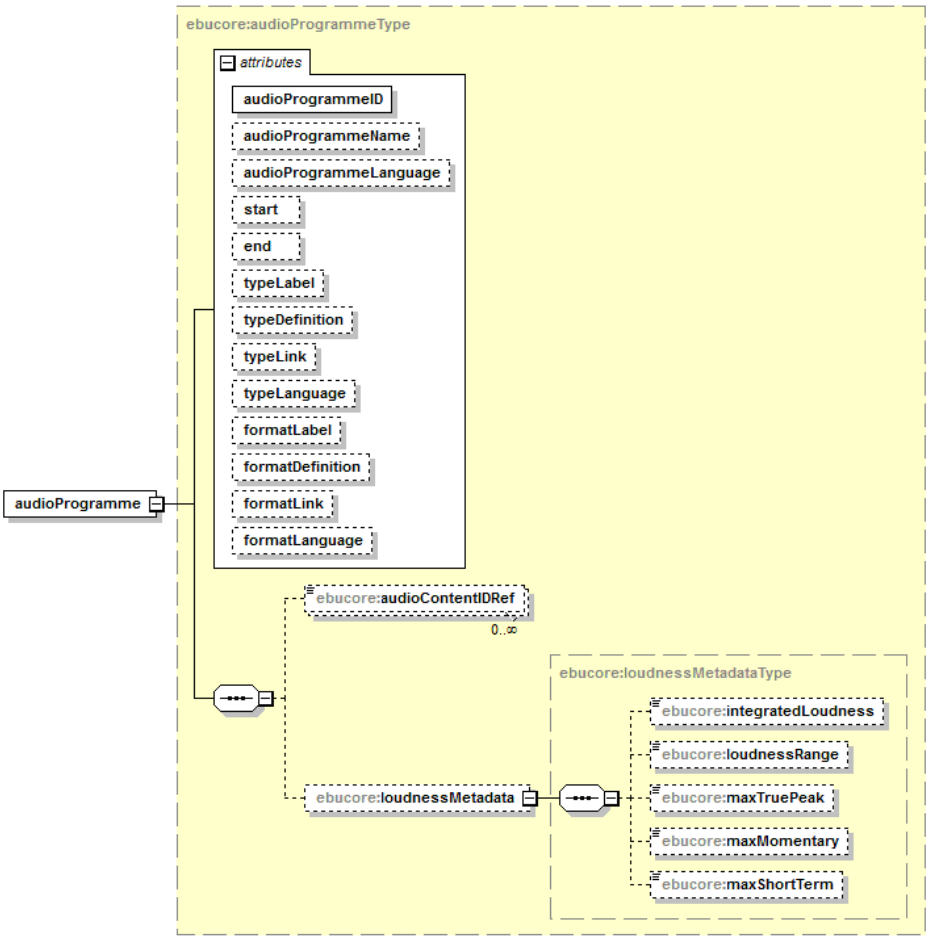


Figure 13: audioProgramme

5.8.1 Attributes

Attribute	Description	Example
audioProgrammeID	ID of the programme	APR_1001
audioProgrammeName	Name of the programme	
audioProgrammeLanguage	Language of the dialogue	fr
start	Start timecode for the programme	00:00:10.00000
end	End timecode for the programme	00:10:00.00000
typeGroup* (Label, Definition, Link, Language)	(not currently used)	
formatGroup* (Label, Definition, Link, Language)	(not currently used)	

5.8.2 Sub-elements

Element	Description	Example
audioContentIDRef	Reference to content	ACO_1001
loudnessMetadata	See Section 5.8.3	

5.8.3 Loudness attributes and sub-elements

Attribute	Description	Example
loudnessMethod	The method or standard used to calculate the loudness metadata	R 128

Element	Description	Units	Example
integratedLoudness	Integrated loudness value	LUFS	-23.0
loudnessRange	Loudness range	LU	10.0
maxTruePeak	Maximum true-peak	dBTP	-2.3
maxMomentary	Maximum momentary loudness	LUFS	-19.0
maxShortTerm	Maximum short term loudness	LUFS	-21.2

Note: For historical reasons, some documents use LKFS as loudness units. Measurements in LUFS and LKFS are identical, but the use of LUFS is in accordance with international standards for the naming of units, whilst that of LKFS is not.

5.8.4 Sample Code

```
<audioProgramme audioProgrammeID="APR_1001" audioProgrammeName="Documentary">
  <audioContentIDRef>ACO_1001</audioContentIDRef>
  <audioContentIDRef>ACO_1002</audioContentIDRef>
</audioProgramme>
```

5.9 audioTrackUID

The audioTrackUID uniquely identifies a track or asset within a file. This element contains information about the bit-depth and sample-rate of the track. It also contains sub-elements that allow the model to be used for non-BWF applications by performing the job of the <*chna*> chunk. When using the model with MXF files the audioMXFLookUp sub-element (which contains sub-elements to refer to the audio essences in the file) is used.

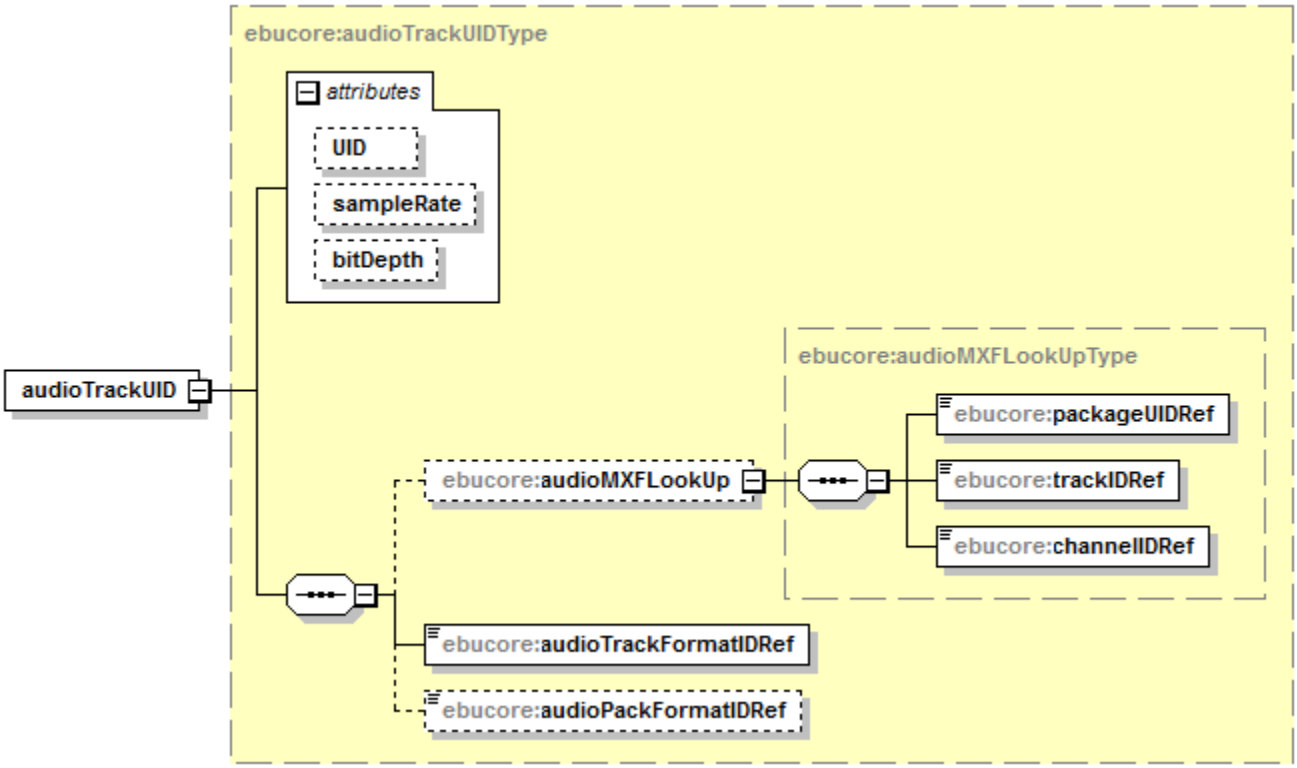


Figure 14: audioTrackUID

5.9.1 Attributes

Attribute	Description	Example
UID	The actual UID value	ATU_00000001
sampleRate	Sample rate of track in Hz	48000
bitDepth	Bit-depth of track in bits	24

5.9.2 Sub-elements

Element	Description	Example
audioMXFLookUp	See Section 5.9.3	
audioTrackFormatIDRef	Reference to an audioTrackFormat description	AT_00010001_01
audioPackFormatIDRef	Reference to an audioPackFormat description	AP_00010002

5.9.3 MXF sub-elements

MXF has different meanings for the terms 'track' and 'channel' from their use in the ADM. In MXF 'track' is the storage medium containing audio or video, and for audio this 'track' can be sub-divided into 'channels'.

Element	Description	Type	Example
packageUIDRef	Reference to an MXF package	UMID string	urn:smpte:umid: 060a2b34.01010105.01010f20.13000000. 540bca53.41434f05.8ce5f4e3.5b72c985
trackIDRef	Reference to an MXF track	int	MXFTRACK_3
channelIDRef	Reference to a channel track	Int	MXFCHAN_1

5.9.4 Sample Code

```
<audioTrackUID UID="ATU_00000001" sampleRate="48000" bitDepth="24"/>
```

5.10 audioFormatExtended

AudioFormatExtended is the parent element, containing all the ADM elements. This will be included in EBU Core Metadata Version 1.5 [4].

5.10.1 Sub-elements

Element	Description
audioProgramme	Description of the whole audio programme.
audioContent	Description of the content of some audio within the programme.
audioObject	The link between the actual audio tracks and their format.
audioPackFormat	A description of a pack of channels that relate together.
audioChannelFormat	A description of an audio channel.
audioStreamFormat	A description of an audio stream.
audioTrackFormat	A description of an audio track.
audioTrackUID	The unique identifier for an actual audio track.

6. Use of IDs

The ID attributes in each of the elements have three main purposes: to allow the elements to reference each other, to provide a unique identification for each defined element, and to provide a logical numerical representation of the contents of the element. The IDs for each element follows the following format:

Element	ID format
audioPackFormat	AP_yyyyxxxx
audioChannelFormat	AC_yyyyxxxx
audioBlockFormat	AB_yyyyxxxx_zzzzzzzz
audioStreamFormat	AS_yyyyxxxx
audioTrackFormat	AT_yyyyxxxx_zz
audioProgramme	APR_wwww
audioContent	ACO_wwww
audioObject	AO_wwww

The yyyy part is a four digit hexadecimal number that represents the **type** of element it is, by using the typeLabel values. Currently there are 5 defined type label values:

typeDefinition	typeLabel	Description
DirectSpeakers	0001	For channel-based audio, where each channel feeds a speaker directly
Matrix	0002	For channel-based audio where channels are matrixed together, such as Mid-Side, Lt/Rt
Objects	0003	For object-based audio where channels represent audio objects (or parts of objects), so include positional information
HOA	0004	For scene-based audio where Ambisonics and HOA are used
Binaural	0005	For binaural audio, where playback is over headphones

The xxxx part is a four digit hexadecimal number which identifies the description within a particular type. Values in the range 0001-0FFF are reserved for standard definition such as 'FrontLeft' or 'Stereo'. Values in the range 1000-FFFF are for custom definitions, which will be particularly used in object-based audio where all the objects will be custom definitions.

In audioBlockFormat the zzzzzzzz part is an 8 digit hexadecimal number that acts as an index/counter for the blocks within the channel. The yyyyxxxx values should match those of the parent audioChannelFormat ID.

In audioTrackFormat the zz part is a 2 digit hexadecimal number that acts as an index/counter for the tracks within the stream. The yyyyxxxx values should match those of the reference audioStreamFormat ID.

The audioProgramme, audioContent and audioObject do not have a type and so have no yyyy values. As there is initially no intention to have standard definitions for these elements the values for wwww will be in the hexadecimal range 1000-FFFF because they will always be custom values. However, keeping the standard range of values (0000-0FFF) set aside for now may be useful in future; for example, EBU R 123 configurations may use them.

7. <chna> Chunk

While the ADM is designed to be a general model beyond application just to the BWF, its relationship with the BWF file is important to explain. The following describes how a BWF file will access the ADM metadata via a new RIFF chunk called <*chna*>. A complete specification of this new chunk will be described in EBU Tech 3285 s7 and so just an overview of its use is given here.

The ADM is linked to the BWF file using the audioTrackFormat, audioPackFormat and audioObject (via audioTrackUID) elements. The BWF file will contain a new chunk called <*chna*> (short for 'channel allocation') which will contain a set of IDs for each track in the file. These IDs will either refer to elements, or be referred to from an element. Each track in the chunk contains the following IDs:

- **audioTrackFormatID** – the ID of the description of a particular audioTrackFormat element. As audioTrackFormat also refers to audioStreamFormat and either audioPackFormat or audioChannelFormat, this ID is enough to describe the format for a particular track.
- **audioPackFormatID** – the ID of the description of a particular audioPackFormat. As most audioChannelFormats need to be assigned to an audioPackFormat (e.g. 'FrontLeft' channel in '5.1' pack), it must be specified in the <*chna*> chunk with this ID.
- **audioTrackUID** – the unique ID that identifies the track. The content descriptor audioObject requires knowledge of which tracks in the file are being described, so contains a list of audioTrackUID references which correspond to audio tracks in the file.

To enable tracks to contain more than one audioTrackFormatID, in order to allow different formats in the track at different times, the track number can be allocated multiple IDs. An example of such as allocation is below:

Track No	audioTrackUID	audioTrackFormatID	audioPackFormatID
1	00000001	00010001_01	00010001
2	00000002	00031001_01	00031001
2	00000003	00031002_01	00031002

Here, track number two has two audioTrackUIDs as the audioTrackFormats and audioPackFormats assigned to it are used at different times in the file. The times of allocation would have to be found be inspecting the audioObject elements that cover those audioTrackUIDs. An example of this is a programme where tracks 1 and 2 contain the theme tune which lasts for the first minute of the file. These tracks are free after this first minute, so some audio objects from the main body of the programme are stored in them subsequently. As the theme tune and the audio objects have completely different formats and contents they require different audioTrackUIDs.

8. Coordinate System

The position elements in audioBlockFormat, for both the 'DirectSpeakers' and 'Objects' typeDefinitions, allow different axes to be specified in the coordinate attribute. The primary coordinate system used is the polar system which uses azimuth, elevation and distance. To ensure consistency when specifying positions each of the polar axes should be based on these guidelines:

- **The origin is in the centre**, where the sweet-spot would be (although some systems do not have a sweet-spot, so the centre of the space should be assumed).
- **Azimuth** – angle in the horizontal plane with 0 degrees as straight ahead, and positive angles to the left (or anti-clockwise) when viewed from above.
- **Elevation** – angle in the vertical plane with 0 degrees horizontally ahead, and positive angles going up.
- **Distance** – a normalised distance, where 1.0 is assumed to be the default radius of the

sphere.

It is also be possible to specify Cartesian coordinates by using X, Y and Z as the coordinate attributes. It is recommended that normalised values be used here, where the values 1.0 and -1.0 are on the surface of the cube, with the origin being the centre of the cube. The direction of each axis should be:

- **X** – left to right, with positive values to the right.
- **Y** – front to back, with positive values to the front.
- **Z** – top to bottom, with positive values to the top.

If normalised distances are used in the coordinate system they can be scaled to an absolute distance by multiplying by the absoluteDistance parameter in the audioPackFormat.

In Ambisonics and HOA, the coordinate system is also Cartesian based, but the axis are different. In this case the direction of each axis is:

- **X** – front to back, with positive values to the front.
- **Y** – left to right, with positive values to the left.
- **Z** – top to bottom, with positive values to the top.

To avoid confusion with the other Cartesian system, it is recommended the axes be labelled 'X_HOA', 'Y_HOA' & 'Z_HOA'. However, the HOA component definitions are unlikely to include coordinate information and so this information is primarily to ensure the rendering is correctly done.

9. References

[1] ITU-R BS.2266, "Framework of future audio broadcasting systems"

[2] ITU-R BS.1909, "Performance requirements for an advanced multichannel stereophonic sound system for use with or without accompanying picture"

[3] EBU Tech 3285, "Specification of the Broadcast Wave Format"

[4] EBU Tech 3293, "EBU Core Metadata Set" (Version 1.5, January 2014)

[5] EBU R 128, "Loudness normalisation and permitted maximum level of audio signals"

[6] EBU R 123, "EBU Audio Track Allocation for File Exchange" (referenced in the annex)

Annex: Examples of ADM usage

This annex contains a selection of examples of metadata that uses the Audio Definition Model. These are to help illustrate how the ADM is used, but should not be considered as references for audio definitions.

A1 Channel-based Example

The most common use of audio is still channel-based, where tracks within a file each represent a static audio channel. This example demonstrates how to define two tracks, streams and channels; and a pack for stereo. The track and stream definitions are for PCM audio. Two objects are defined, both stereo, but containing different content so there are 4 tracks used. This example uses a programme called 'Documentary' containing 'Music' and 'Speech' each defined as separate stereo objects.

The format-related elements in this example represent a tiny subset of the standard reference set of definitions. In practice, this XML code would be part of the standard reference file and would not have to be included in the BWF file. All that would be required is a <*chna*> chunk with the references to the audioTrackFormats and audioPackFormats and any extra XML required for audioObject, audioContent and audioProgramme.

A1.1 Summary of Elements

These are the elements in the format part of the description:

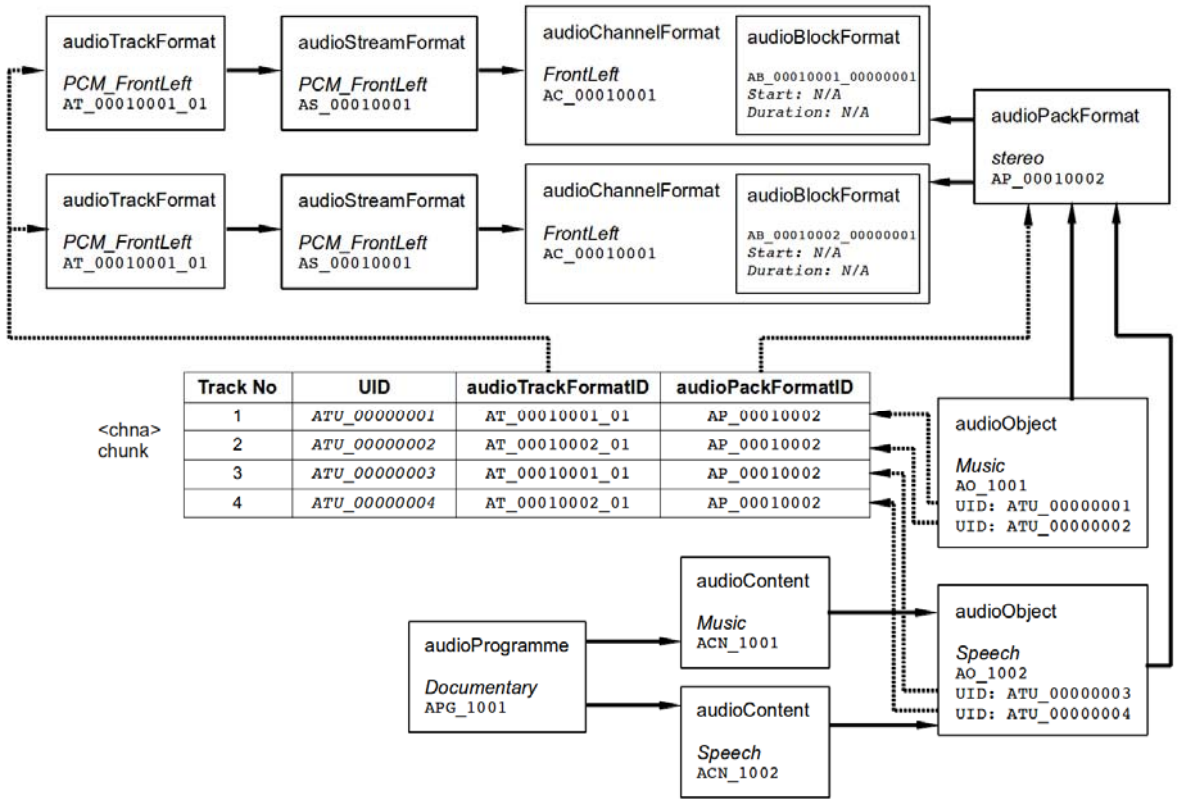
Element	ID	Name	Description
audioTrackFormat	AT_00010001_01	PCM_FrontLeft	Defines track as PCM
audioTrackFormat	AT_00010002_01	PCM_FrontRight	Defines track as PCM
audioStreamFormat	AS_00010001	PCM_FrontLeft	Defines stream as PCM
audioStreamFormat	AS_00010002	PCM_FrontRight	Defines stream as PCM
audioChannelFormat & audioBlockFormat	AC_00010001 AB_00010001_00000001	FrontLeft	Describes channel as front left with a position and speaker reference
audioChannelFormat & audioBlockFormat	AC_00010002 AB_00010002_00000001	FrontRight	Describes channel as front right with a position and speaker reference
audioPackFormat	AP_00010002	Stereo	Defines a stereo pack referring to two channels.

These are the elements in the content part of the description:

Element	ID	Name	Description
audioObject	AO_1001	Music	Object for 'Music', stereo format
audioObject	AO_1002	Speech	Object for 'Speech', stereo format
audioContent	ACN_1001	Music	Music content
audioContent	ACN_1002	Speech	Speech content
audioProgramme	APG_1001	Documentary	Programme 'Documentary' containing 'Music' and 'Speech' content

A1.2 Diagram

The diagram shows how the defined elements relate to each other. The top half of the diagram covers the elements that describe the 2 channel stereo format. The <*chna*> chunk in the middle shows how the four tracks are connected to the format definitions. The content definition elements are at the bottom of the diagram, with the audioObject elements containing the track UID references to the UIDs in the <chna> chunk.



A1.3 Sample Code

This XML sample code does not include the audioFormatExtended parent element and the XML header for clarity.

The first excerpt of code covers the format elements, which could be contained within the standard reference file:

```
<!-- ############ -->
<!-- PACKS -->
<!-- ############ -->

<audioPackFormat audioPackFormatID="AP_00010002" audioPackFormatName="Stereo"
typeLabel="0001"  typeDefinition="DirectSpeakers">
  <audioChannelFormatIDRef>AC_00010001</audioChannelFormatIDRef>
  <audioChannelFormatIDRef>AC_00010002</audioChannelFormatIDRef>
</audioPackFormat>

<!-- ############ -->
<!-- CHANNELS -->
<!-- ############ -->
```

```
<audioChannelFormat audioChannelFormatID="AC_00010001" audioChannelFormatName="FrontLeft"
typeLabel="0001" typeDefinition="DirectSpeakers">
  <audioBlockFormat audioBlockFormatID="AB_00010001_00000001">
    <speakerLabel>M+30</speakerLabel>
    <position coordinate="azimuth">30.0</position>
    <position coordinate="elevation">0.0</position>
    <position coordinate="distance">1.0</position>
  </audioBlockFormat>
</audioChannelFormat>

<audioChannelFormat audioChannelFormatID="AC_00010002" audioChannelFormatName="FrontRight"
typeLabel="0001" typeDefinition="DirectSpeakers">
  <audioBlockFormat audioBlockFormatID="AB_00010002_00000001">
    <speakerLabel>M-30</speakerLabel>
    <position coordinate="azimuth">-30.0</position>
    <position coordinate="elevation">0.0</position>
    <position coordinate="distance">1.0</position>
  </audioBlockFormat>
</audioChannelFormat>

<!-- ############ -->
<!-- STREAMS -->
<!-- ############ -->

<audioStreamFormat audioStreamFormatID="AS_00010001" audioStreamFormatName="PCM_FrontLeft"
formatLabel="0001" formatDefinition="PCM">
  <audioChannelFormatIDRef>AC_00010001</audioChannelFormatIDRef>
  <audioTrackFormatIDRef>AS_00010001_AT_01</audioTrackFormatIDRef>
</audioStreamFormat>

<audioStreamFormat audioStreamFormatID="AS_00010002" audioStreamFormatName="PCM_FrontRight"
formatLabel="0001" formatDefinition="PCM">
  <audioChannelFormatIDRef>AC_00010002</audioChannelFormatIDRef>
  <audioTrackFormatIDRef>AS_00010002_AT_01</audioTrackFormatIDRef>
</audioStreamFormat>

<!-- ############ -->
<!-- AUDIO TRACKS -->
<!-- ############ -->

<audioTrackFormat audioTrackFormatID="AS_00010001_AT_01"
audioTrackFormatName="PCM_FrontLeft"  formatLabel="0001" formatDefinition="PCM">
  <audioStreamFormatIDRef>AS_00010001</audioStreamFormatIDRef>
</audioTrackFormat>

<audioTrackFormat audioTrackFormatID="AS_00010002_AT_01"
audioTrackFormatName="PCM_FrontRight"  formatLabel="0001" formatDefinition="PCM">
  <audioStreamFormatIDRef>AS_00010002</audioStreamFormatIDRef>
</audioTrackFormat>
```

The second excerpt covers the content part, which would have to be included in the <axml> chunk of the BWF file:

```
<!-- ############ -->
<!-- PROGRAMMES -->
<!-- ############ -->

<audioProgramme audioProgrammeID="APG_1001" audioProgrammeName="Documentary">
  <audioContentIDRef>ACN_1001</audioContentIDRef>
  <audioContentIDRef>ACN_1002</audioContentIDRef>
</audioProgramme>

<!-- ############ -->
<!-- CONTENTS -->
<!-- ############ -->

<audioContent audioContentID="ACN_1001" audioContentName="Music">
  <audioObjectIDRef>AO_1001</audioObjectIDRef>
  <loudnessMetadata>
    <integratedLoudness>-28.0</integratedLoudness>
  </loudnessMetadata>
</audioContent>

<audioContent audioContentID="ACN_1002" audioContentName="Speech">
  <audioObjectIDRef>AO_1002</audioObjectIDRef>
  <loudnessMetadata>
    <integratedLoudness>-23.0</integratedLoudness>
  </loudnessMetadata>
</audioContent>

<!-- ############ -->
<!-- OBJECTS -->
<!-- ############ -->

<audioObject audioObjectID="AO_1001" audioObjectName="Music" start="00:00:00.00">
  <audioPackFormatIDRef>AP_00010002</audioPackFormatIDRef>
  <audioTrackUIDRef>ATU_00000001</audioTrackUIDRef>
  <audioTrackUIDRef>ATU_00000002</audioTrackUIDRef>
</audioObject>

<audioObject audioObjectID="AO_1002" audioObjectName="Speech" start="00:00:00.00">
  <audioPackFormatIDRef>AP_00010002</audioPackFormatIDRef>
  <audioTrackUIDRef>ATU_00000003</audioTrackUIDRef>
  <audioTrackUIDRef>ATU_00000004</audioTrackUIDRef>
</audioObject>
```

A2 Object-based Example

To demonstrate how the ADM can be used in object-based audio here is a simple example using a single object. This example uses multiple audioBlockFormats within an audioChannelFormat to describe the dynamic properties of an object called “Car”. The audioBlockFormats uses the start and duration attributes to frame the time dependent metadata, thus allowing the object’s position to move in space.

A2.1 Summary of Elements

These are the elements in the format part of the description:

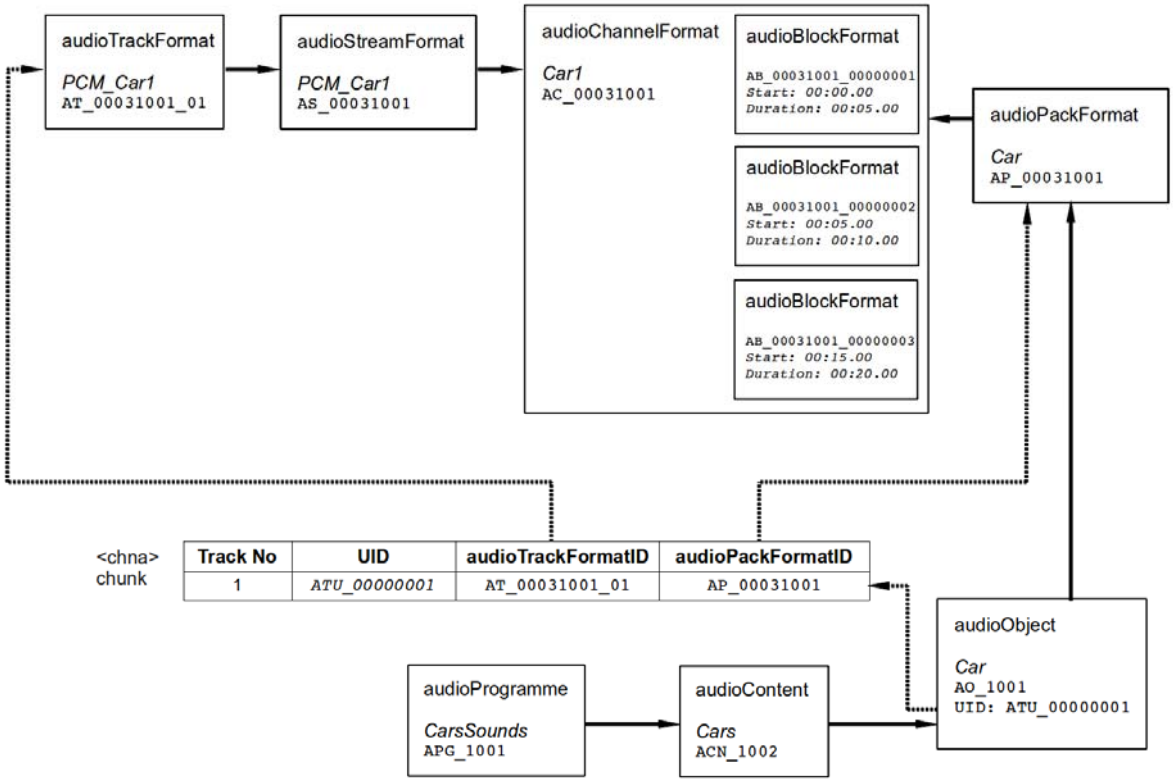
Element	ID	Name	Description
audioTrackFormat	AT_00031001_01	PCM_Car1	Defines track as PCM
audioStreamFormat	AS_00031001	PCM_Car1	Defines stream as PCM
audioChannelFormat & audioBlockFormat	AC_00031001 AB_00031001_00000001 AB_00031001_00000002 AB_00031001_00000003	Car1	Describes channel as an object type containing 3 blocks with different positional metadata in each.
audioPackFormat	AP_00031001	Car	Defines a pack referring to one channel.

These are the elements in the content part of the description:

Element	ID	Name	Description
audioObject	AO_1001	Car	Object for 'Car, stereo format
audioContent	ACN_1001	Cars	'Cars' content
audioProgramme	APG_1001	CarsSounds	Programme 'CarsSounds' containing 'Cars' content

A2.2 Diagram

The diagram shows how the defined elements relate to each other. The top half of the diagram covers the elements that describe the single channel object containing 3 blocks. The <*chna*> chunk in the middle shows how the single track is connected to the format definitions. The content definition elements are at the bottom of the diagram, with the audioObject element containing the track UID references to the UID in the <*chna*> chunk.



A2.3 Sample Code

This XML sample code does not include the audioFormatExtended parent element and the XML header for clarity. The excerpt of code covers both the format and content elements:

```
<!-- ############ -->
<!-- PROGRAMMES -->
<!-- ############ -->

<audioProgramme audioProgrammeID="APG_1001" audioProgrammeName="CarsSounds">
  <audioContentIDRef>ACN_1001</audioContentIDRef>
</audioProgramme>
```

```
<!-- ############ -->
<!-- CONTENTS -->
<!-- ############ -->

<audioContent audioContentID="ACN_1001" audioContentName="Cars">
  <audioObjectIDRef>AO_1001</audioObjectIDRef>
  <loudnessMetadata>
    <integratedLoudness>-23.0</integratedLoudness>
  </loudnessMetadata>
</audioContent>

<!-- ############ -->
<!-- OBJECTS -->
<!-- ############ -->

<audioObject audioObjectID="AO_1001" audioObjectName="Car" start="00:00:00.00000">
  <audioPackFormatIDRef>AP_00031001</audioPackFormatIDRef>
  <audioTrackUIDRef>ATU_00000001</audioTrackUIDRef>
</audioObject>

<!-- ############ -->
<!-- PACKS -->
<!-- ############ -->

<audioPackFormat audioPackFormatID="AP_00031001" audioPackFormatName="Car" typeLabel="0003"
typeDefinition="Objects">
  <audioChannelFormatIDRef>AC_00031001</audioChannelFormatIDRef>
</audioPackFormat>

<!-- ############ -->
<!-- CHANNELS -->
<!-- ############ -->

<audioChannelFormat audioChannelFormatID="AC_00031001" audioChannelFormatName="Car1"
typeLabel="0003"  typeDefinition="Objects">
  <audioBlockFormat audioBlockFormatID="AB_00031001_00000001" rtime="00:00:00.00000"
duration="00:00:05.00000">
    <position coordinate="azimuth">-22.5</position>
    <position coordinate="elevation">5.0</position>
    <position coordinate="distance">1.0</position>
  </audioBlockFormat>
  <audioBlockFormat audioBlockFormatID="AB_00031001_00000002" rtime="00:00:05.00000"
duration="00:00:10.00000">
    <position coordinate="azimuth">-24.5</position>
    <position coordinate="elevation">6.0</position>
    <position coordinate="distance">0.9</position>
  </audioBlockFormat>
  <audioBlockFormat audioBlockFormatID="AB_00031001_00000003" rtime="00:00:15.00000"
duration="00:00:20.00000">
    <position coordinate="azimuth">-26.5</position>
    <position coordinate="elevation">7.0</position>
    <position coordinate="distance">0.8</position>
  </audioBlockFormat>
</audioChannelFormat>


<!-- ############ -->
<!-- STREAMS -->
<!-- ############ -->
```

```
<audioStreamFormat audioStreamFormatID="AS_00031001" audioStreamFormatName="PCM_Car1"
formatLabel="0001" formatDefinition="PCM">
  <audioChannelFormatIDRef>AC_00031001</audioChannelFormatIDRef>
  <audioTrackFormatIDRef>AS_00031001_AT_01</audioTrackFormatIDRef>
</audioStreamFormat>

<!-- ############ -->
<!-- AUDIO TRACKS -->
<!-- ############ -->

<audioTrackFormat audioTrackFormatID="AS_00031001_AT_01" audioTrackFormatName="PCM_Car1"
formatLabel="0001" formatDefinition="PCM">
  <audioStreamFormatIDRef>AS_00031001</audioStreamFormatIDRef>
</audioTrackFormat>
```

A3 Scene-based Example

The other main type of audio is scene-based where the audio channels are representing Ambisonic/HOA components. Their use is very similar to that of the channel-based approach with the main difference being the parameters used within audioBlockFormat. This example shows a simple 1st order Ambisonic (using the N3D method) configuration using 4 channels mapped onto 4 tracks. Like the channel-based approach, the format elements would be defined in a standard reference file so in practice would not need to be included in the BWF file itself.

A3.1 Summary of Elements

These are the elements in the format part of the description:

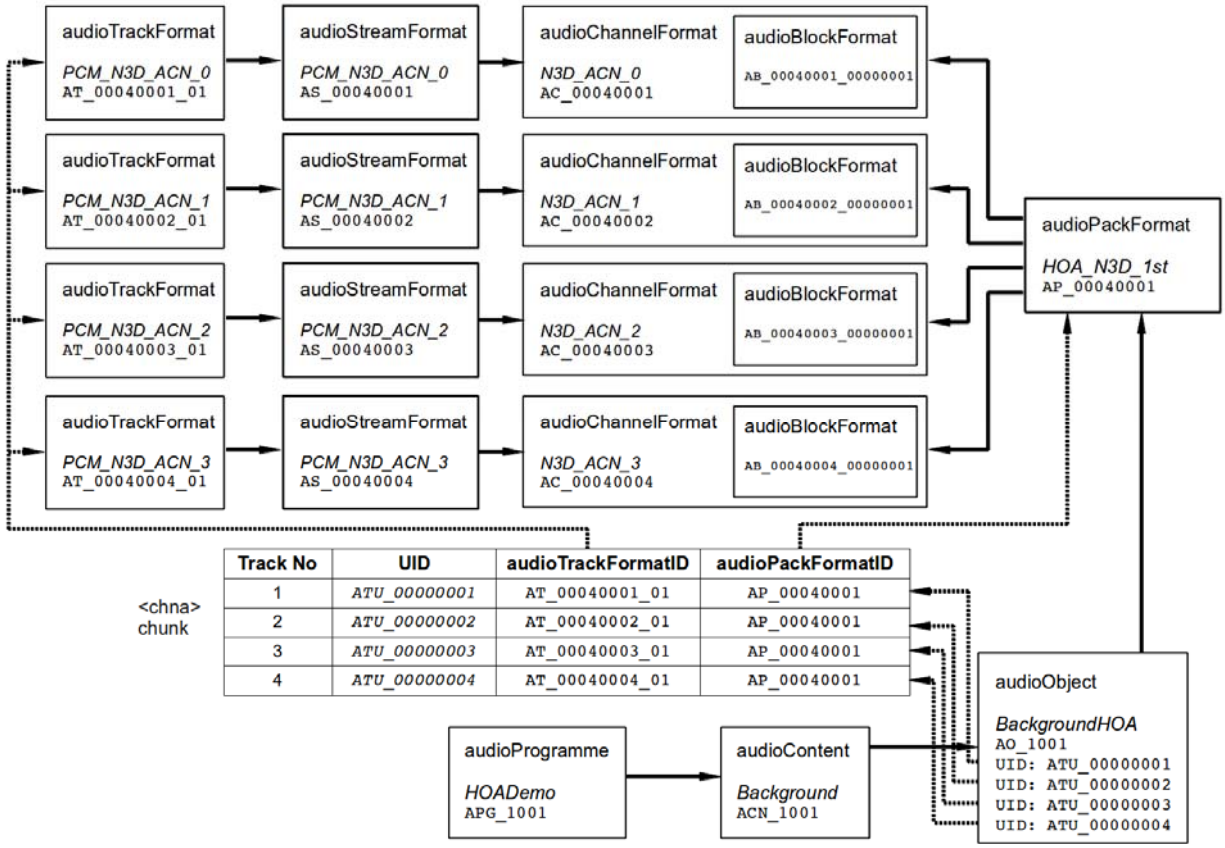
Element	ID	Name	Description
audioTrackFormat	AT_00040001_01	PCM_N3D_ACN_0	Defines track as PCM
audioTrackFormat	AT_00040002_01	PCM_N3D_ACN_1	Defines track as PCM
audioTrackFormat	AT_00040003_01	PCM_N3D_ACN_2	Defines track as PCM
audioTrackFormat	AT_00040004_01	PCM_N3D_ACN_3	Defines track as PCM
audioStreamFormat	AS_00040001	PCM_N3D_ACN_0	Defines stream as PCM
audioStreamFormat	AS_00040002	PCM_N3D_ACN_1	Defines stream as PCM
audioStreamFormat	AS_00040003	PCM_N3D_ACN_2	Defines stream as PCM
audioStreamFormat	AS_00040004	PCM_N3D_ACN_3	Defines stream as PCM
audioChannelFormat & audioBlockFormat	AC_00040001 AB_00040001_00000001	N3D_ACN_0	Describes channel as ACN0 HOA component
audioChannelFormat & audioBlockFormat	AC_00040002 AB_00040002_00000001	N3D_ACN_1	Describes channel as ACN1 HOA component
audioChannelFormat & audioBlockFormat	AC_00040003 AB_00040003_00000001	N3D_ACN_2	Describes channel as ACN2 HOA component
audioChannelFormat & audioBlockFormat	AC_00040004 AB_00040004_00000001	N3D_ACN_3	Describes channel as ACN3 HOA component
audioPackFormat	AP_00040001	HOA_N3D_1st	Defines a 1st order HOA pack referring to four ACN channels.

These are the elements in the content part of the description:

Element	ID	Name	Description
audioObject	AO_1001	BackgroundHOA	Object for 'BackgroundHOA', 1st order HOA format
audioContent	ACN_1001	Background	'Background' content
audioProgramme	APG_1001	HOADemo	'HOADemo' containing a 'Background'content

A3.2 Diagram

The diagram shows how the defined elements relate to each other. The top half of the diagram covers the elements that describe the 4 channels of the 1st order HOA (N3D method). The *<chna>* chunk in the middle shows how the four tracks are connected to the format definitions. The content definition elements are at the bottom of the diagram, with the audioObject element containing the track UID references to the UIDs in the <chna> chunk.



A3.3 Sample Code

This XML sample code does not include the audioFormatExtended parent element and the XML header for clarity. The first excerpt of code covers the format elements, which could be contained within the standard reference file:

```
<!-- ############ -->
<!-- PACKS -->
<!-- ############ -->

<audioPackFormat audioPackFormatID="AP_00040001" audioPackFormatName="HOA_N3D_1st"
typeLabel="0004"  typeDefinition="HOA">
  <audioChannelFormatIDRef>AC_00040001</audioChannelFormatIDRef>
```

```
  <audioChannelFormatIDRef>AC_00040002</audioChannelFormatIDRef>
  <audioChannelFormatIDRef>AC_00040003</audioChannelFormatIDRef>
  <audioChannelFormatIDRef>AC_00040004</audioChannelFormatIDRef>
</audioPackFormat>

<!-- ############ -->
<!-- CHANNELS -->
<!-- ############ -->

<audioChannelFormat audioChannelFormatID="AC_00040001" audioChannelFormatName="N3D_ACN_0"
typeDefinition="HOA">
  <audioBlockFormat audioBlockFormatID="AB_00040001_00000001">
    <equation>1</equation>
    <degree>0</degree>
    <order>0</order>
  </audioBlockFormat>
</audioChannelFormat>

<audioChannelFormat audioChannelFormatID="AC_00040002" audioChannelFormatName="N3D_ACN_1"
typeDefinition="HOA">
  <audioBlockFormat audioBlockFormatID="AB_00040002_00000001">
    <equation>sqrt(3)*cos(E)</equation>
    <degree>1</degree>
    <order>-1</order>
  </audioBlockFormat>
</audioChannelFormat>

<audioChannelFormat audioChannelFormatID="AC_00040003" audioChannelFormatName="N3D_ACN_2"
typeDefinition="HOA">
  <audioBlockFormat audioBlockFormatID="AB_00040003_00000001">
    <equation>sqrt(3)*sin(E)</equation>
    <degree>1</degree>
    <order>0</order>
  </audioBlockFormat>
</audioChannelFormat>

<audioChannelFormat audioChannelFormatID="AC_00040004" audioChannelFormatName="N3D_ACN_3"
typeDefinition="HOA">
  <audioBlockFormat audioBlockFormatID="AB_00040004_00000001">
    <equation>sqrt(3)*cos(E)*cos(A)</equation>
    <degree>1</degree>
    <order>1</order>
  </audioBlockFormat>
</audioChannelFormat>

<!-- ############ -->
<!-- STREAMS -->
<!-- ############ -->

<audioStreamFormat audioStreamFormatID="AS_00040001" audioStreamFormatName="PCM_N3D_ACN_0"
formatLabel="0001" formatDefinition="PCM">
  <audioChannelFormatIDRef>AC_00040001</audioChannelFormatIDRef>
  <audioTrackFormatIDRef>AS_00040001_AT_01</audioTrackFormatIDRef>
</audioStreamFormat>

<audioStreamFormat audioStreamFormatID="AS_00040002" audioStreamFormatName="PCM_N3D_ACN_1"
formatLabel="0001" formatDefinition="PCM">
  <audioChannelFormatIDRef>AC_00040002</audioChannelFormatIDRef>
  <audioTrackFormatIDRef>AS_00040002_AT_01</audioTrackFormatIDRef>
</audioStreamFormat>
```

```
<audioStreamFormat audioStreamFormatID="AS_00040003" audioStreamFormatName="PCM_N3D_ACN_2"
formatLabel="0001" formatDefinition="PCM">
  <audioChannelFormatIDRef>AC_00040003</audioChannelFormatIDRef>
  <audioTrackFormatIDRef>AS_00040003_AT_01</audioTrackFormatIDRef>
</audioStreamFormat>

<audioStreamFormat audioStreamFormatID="AS_00040004" audioStreamFormatName="PCM_N3D_ACN_3"
formatLabel="0001" formatDefinition="PCM">
  <audioChannelFormatIDRef>AC_00040004</audioChannelFormatIDRef>
  <audioTrackFormatIDRef>AS_00040004_AT_01</audioTrackFormatIDRef>
</audioStreamFormat>

<!-- ############ -->
<!-- AUDIO TRACKS -->
<!-- ############ -->

<audioTrackFormat audioTrackFormatID="AS_00040001_AT_01"
audioTrackFormatName="PCM_N3D_ACN_0"  formatLabel="0001" formatDefinition="PCM">
  <audioStreamFormatIDRef>AS_00040001</audioStreamFormatIDRef>
</audioTrackFormat>

<audioTrackFormat audioTrackFormatID="AS_00040002_AT_01"
audioTrackFormatName="PCM_N3D_ACN_1"  formatLabel="0001" formatDefinition="PCM">
  <audioStreamFormatIDRef>AS_00040002</audioStreamFormatIDRef>
</audioTrackFormat>

<audioTrackFormat audioTrackFormatID="AS_00040003_AT_01"
audioTrackFormatName="PCM_N3D_ACN_2"  formatLabel="0001" formatDefinition="PCM">
  <audioStreamFormatIDRef>AS_00040003</audioStreamFormatIDRef>
</audioTrackFormat>

<audioTrackFormat audioTrackFormatID="AS_00040004_AT_01"
audioTrackFormatName="PCM_N3D_ACN_3"  formatLabel="0001" formatDefinition="PCM">
  <audioStreamFormatIDRef>AS_00040004</audioStreamFormatIDRef>
</audioTrackFormat>
```

The second excerpt covers the content part, which would have to be included in the <axml> chunk of the BWF file:

```
<!-- ############ -->
<!-- PROGRAMMES -->
<!-- ############ -->

<audioProgramme audioProgrammeID="APG_1001" audioProgrammeName="HOADemo">
  <audioContentIDRef>ACN_1001</audioContentIDRef>
</audioProgramme>

<!-- ############ -->
<!-- CONTENTS -->
<!-- ############ -->

<audioContent audioContentID="ACN_1001" audioContentName="Background">
  <audioObjectIDRef>AO_1001</audioObjectIDRef>
</audioContent>

<!-- ############ -->
<!-- OBJECTS -->
```

```
<!-- ############ -->

<audioObject audioObjectID="AO_1001" audioObjectName="BackgroundHOA">
  <audioPackFormatIDRef>AP_00040001</audioPackFormatIDRef>
  <audioTrackUIDRef>ATU_00000001</audioTrackUIDRef>
  <audioTrackUIDRef>ATU_00000002</audioTrackUIDRef>
  <audioTrackUIDRef>ATU_00000003</audioTrackUIDRef>
  <audioTrackUIDRef>ATU_00000004</audioTrackUIDRef>
</audioObject>
```

A4 MXF Mapping Example

The ADM has been designed to not only allow BWF files to become a flexible multichannel file format, but also be incorporated by other file formats. Currently, MXF (Material Exchange Format – SMPTE 377M), which carries both video and audio, has a rather limited capability in terms of specifying its audio format. The ADM could be used by MXF files in a similar way to BWF files allowing a comprehensive format description of the audio.

MXF files often use EBU R 123 [6](“EBU Audio Track Allocation for File Exchange”) audio track configurations. This is a set of channel and matrix-based track allocations for between 2 and 16 track files or streams. This example will show how a particular R123 configuration can be represented by the ADM that’s suitable for MXF.

This example will demonstrate how the 4a R 123 configuration can be represented by the ADM. This configuration uses 4 tracks:

Track Number	Track Use	Group
1	Stereo Left (PCM)	PCM Stereo pair
2	Stereo Right (PCM)	
3	MCA (Dolby E)	Multichannel audio Dolby E stream
4	MCA (Dolby E)	

A4.1 Summary of Elements

These are the elements in the format part of the description:

Element	ID	Name	Description
audioTrackFormat	AT_00010001_01	PCM_FrontLeft	Defines track as PCM
audioTrackFormat	AT_00010002_01	PCM_FrontRight	Defines track as PCM
audioTrackFormat	AT_00020001_01	DolbyE1	Defines track as containing Dolby E data
audioTrackFormat	AT_00020001_02	DolbyE1	Defines track as containing Dolby E data
audioStreamFormat	AS_00010001	PCM_FrontLeft	Defines stream as PCM
audioStreamFormat	AS_00010002	PCM_FrontRight	Defines stream as PCM
audioStreamFormat	AS_00020001	DolbyE_5.1	Defines stream as Dolby E
audioChannelFormat & audioBlockFormat	AC_00010001 AB_00010001_00000001	FrontLeft	Describes channel as front left with a position and speaker reference
audioChannelFormat & audioBlockFormat	AC_00010002 AB_00010002_00000001	FrontRight	Describes channel as front right with a position and speaker reference
audioChannelFormat & audioBlockFormat	AC_00010003 AB_00010003_00000001	FrontCentre	Describes channel as front centre with a position and speaker reference
audioChannelFormat & audioBlockFormat	AC_00010004 AB_00010004_00000001	LFE	Describes channel as LFE with a position and speaker reference
audioChannelFormat & audioBlockFormat	AC_00010005 AB_00010005_00000001	SurroundLeft	Describes channel as front right with a position and speaker reference
audioChannelFormat & audioBlockFormat	AC_00010006 AB_00010006_00000001	SurroundRight	Describes channel as front right with a position and speaker reference
audioPackFormat	AP_00010002	Stereo	Defines a stereo pack referring to two channels.
audioPackFormat	AP_00010003	5.1	Defines a 5.1 pack referring to 6 channels.

These are the elements in the content part of the description:

Element	ID	Name	Description
audioObject	AO_0041	R123_4a	Object for R123 4a configuration
audioObject	AO_0002	R123_Stereo	Object for stereo
audioObject	AO_0004	R123_5.1	Object for 5.1

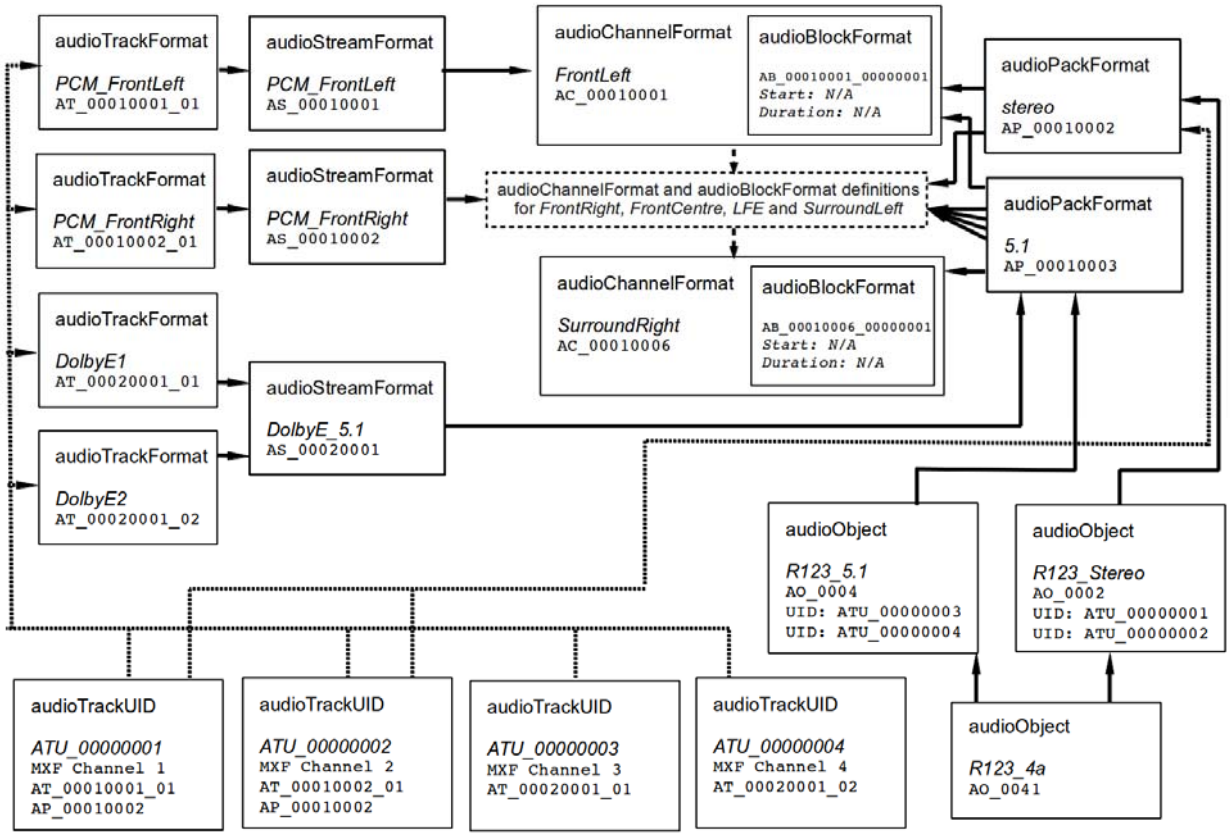
A4.2 Diagram

The diagram shows how the defined elements relate to each other. The top half of the diagram covers the elements that describe the 2 channel stereo PCM format and the 6 channel Dolby E 5.1 encoded format. In the Dolby E part, two audioTrackFormats refer to a single audioStreamFormat as Dolby E requires that two tracks are combined to decode the audio signals. The Dolby E audioStreamFormat refers to an audioPackFormat as it is representing a group of channels rather than a single one. This 5.1 audioPackFormat refers to the 6 audioChannelFormats that describe each channel.

The R123 4a configuration is represented by an audioObject (named 'R123_4a') which refers to two further audioObjects (for the stereo and 5.1 groups) which contain the references to the audioTrackUIDs. This demonstrates the nesting feature of audioObjects.

As MXF does not feature a <*chna*> chunk, it uses sub-elements of audioTrackUID to generate

references to the essences within the MXF file. The audioMXFLookUp sub-element is designed to facilitate these relationships.



A4.3 Sample Code

This XML sample code does not include the audioFormatExtended parent element and the XML header for clarity.

The first excerpt of code covers the format elements, which could be contained within the standard reference file:

```
<!-- ############ -->
<!-- PACKS -->
<!-- ############ -->

<audioPackFormat audioPackFormatID="AP_00010002" audioPackFormatName="Stereo"
typeLabel="0001"  typeDefinition="DirectSpeakers">
  <audioChannelFormatIDRef>AC_00010001</audioChannelFormatIDRef>
  <audioChannelFormatIDRef>AC_00010002</audioChannelFormatIDRef>
</audioPackFormat>

<audioPackFormat audioPackFormatID="AP_00010003" audioPackFormatName="5.1" typeLabel="0001"
typeDefinition="DirectSpeakers">
  <audioChannelFormatIDRef>AC_00010001</audioChannelFormatIDRef>
  <audioChannelFormatIDRef>AC_00010002</audioChannelFormatIDRef>
  <audioChannelFormatIDRef>AC_00010003</audioChannelFormatIDRef>
  <audioChannelFormatIDRef>AC_00010004</audioChannelFormatIDRef>
  <audioChannelFormatIDRef>AC_00010005</audioChannelFormatIDRef>
```

```
  <audioChannelFormatIDRef>AC_00010006</audioChannelFormatIDRef>
</audioPackFormat>

<!-- ############ -->
<!-- CHANNELS -->
<!-- ############ -->

<audioChannelFormat audioChannelFormatID="AC_00010001" audioChannelFormatName="FrontLeft"
typeLabel="0001" typeDefinition="DirectSpeakers">
  <audioBlockFormat audioBlockFormatID="AB_00010001_00000001">
    <speakerLabel>M+30</speakerLabel>
    <position coordinate="azimuth">30.0</position>
    <position coordinate="elevation">0.0</position>
    <position coordinate="distance">1.0</position>
  </audioBlockFormat>
</audioChannelFormat>

<audioChannelFormat audioChannelFormatID="AC_00010002" audioChannelFormatName="FrontRight"
typeLabel="0001" typeDefinition="DirectSpeakers">
  <audioBlockFormat audioBlockFormatID="AB_00010002_00000001">
    <speakerLabel>M-30</speakerLabel>
    <position coordinate="azimuth">-30.0</position>
    <position coordinate="elevation">0.0</position>
    <position coordinate="distance">1.0</position>
  </audioBlockFormat>
</audioChannelFormat>

<audioChannelFormat audioChannelFormatID="AC_00010003" audioChannelFormatName="FrontCentre"
typeLabel="0001" typeDefinition="DirectSpeakers">
  <audioBlockFormat audioBlockFormatID="AB_00010003_00000001">
    <speakerLabel>M+00</speakerLabel>
    <position coordinate="azimuth">0.0</position>
    <position coordinate="elevation">0.0</position>
    <position coordinate="distance">1.0</position>
  </audioBlockFormat>
</audioChannelFormat>

<audioChannelFormat audioChannelFormatID="AC_00010004" audioChannelFormatName="LFE"
typeLabel="0001"  typeDefinition="DirectSpeakers">
  <frequency typeDefinition="lowPass">200</frequency>
  <audioBlockFormat audioBlockFormatID="AB_00010004_00000001">
    <speakerLabel>LFE+00</speakerLabel>
    <position coordinate="azimuth">0.0</position>
    <position coordinate="elevation">-20.0</position>
    <position coordinate="distance">1.0</position>
  </audioBlockFormat>
</audioChannelFormat>

<audioChannelFormat audioChannelFormatID="AC_00010005"
audioChannelFormatName="SurroundLeft"  typeLabel="0001" typeDefinition="DirectSpeakers">
  <audioBlockFormat audioBlockFormatID="AB_00010005_00000001">
    <speakerLabel>M+110</speakerLabel>
    <position coordinate="azimuth">110.0</position>
    <position coordinate="elevation">0.0</position>
    <position coordinate="distance">1.0</position>
  </audioBlockFormat>
</audioChannelFormat>

<audioChannelFormat audioChannelFormatID="AC_00010006"
audioChannelFormatName="SurroundRight"  typeLabel="0001" typeDefinition="DirectSpeakers">
```

```
  <audioBlockFormat audioBlockFormatID="AB_00010006_00000001">
    <speakerLabel>M-110</speakerLabel>
    <position coordinate="azimuth">-110.0</position>
    <position coordinate="elevation">0.0</position>
    <position coordinate="distance">1.0</position>
  </audioBlockFormat>
</audioChannelFormat>

<!-- ############ -->
<!-- STREAMS -->
<!-- ############ -->

<audioStreamFormat audioStreamFormatID="AS_00010001" audioStreamFormatName="PCM_FrontLeft"
formatLabel="0001" formatDefinition="PCM">
  <audioChannelFormatIDRef>AC_00010001</audioChannelFormatIDRef>
  <audioTrackFormatIDRef>AS_00010001_AT_01</audioTrackFormatIDRef>
</audioStreamFormat>

<audioStreamFormat audioStreamFormatID="AS_00010002" audioStreamFormatName="PCM_FrontRight"
formatLabel="0001" formatDefinition="PCM">
  <audioChannelFormatIDRef>AC_00010002</audioChannelFormatIDRef>
  <audioTrackFormatIDRef>AS_00010002_AT_01</audioTrackFormatIDRef>
</audioStreamFormat>

<audioStreamFormat audioStreamFormatID="AS_00020001" audioStreamFormatName="DolbyE_5.1"
formatLabel="DolbyE" formatDefinition="DolbyE">
  <audioPackFormatIDRef>AP_00010003</audioPackFormatIDRef>
  <audioTrackFormatIDRef>AS_00020001_AT_01</audioTrackFormatIDRef>
  <audioTrackFormatIDRef>AS_00020001_AT_02</audioTrackFormatIDRef>
</audioStreamFormat>



<!-- ############ -->
<!-- AUDIO TRACKS -->
<!-- ############ -->

<audioTrackFormat audioTrackFormatID="AS_00010001_AT_01"
audioTrackFormatName="PCM_FrontLeft"  formatLabel="0001" formatDefinition="PCM">
  <audioStreamFormatIDRef>AS_00010001</audioStreamFormatIDRef>
</audioTrackFormat>

<audioTrackFormat audioTrackFormatID="AS_00010002_AT_01"
audioTrackFormatName="PCM_FrontRight"  formatLabel="0001" formatDefinition="PCM">
  <audioStreamFormatIDRef>AS_00010002</audioStreamFormatIDRef>
</audioTrackFormat>

<audioTrackFormat audioTrackFormatID="AS_00020001_AT_01" audioTrackFormatName="DolbyE1"
formatLabel="0002" formatDefinition="data">
  <audioStreamFormatIDRef>AS_00020001</audioStreamFormatIDRef>
</audioTrackFormat>

<audioTrackFormat audioTrackFormatID="AS_00020001_AT_02" audioTrackFormatName="DolbyE2"
formatLabel="0002" formatDefinition="data">
  <audioStreamFormatIDRef>AS_00020001</audioStreamFormatIDRef>
</audioTrackFormat>
```

The second excerpt (below) covers the content part, in this case audioObjects and audioTrackUIDs, which should be contained within the MXF file. The audioTrackUIDs contain the audioMXFLoopUp elements which locate the essence within the MXF file.

```
<!-- ############ -->
<!-- OBJECTS -->
<!-- ############ -->

<audioObject audioObjectID="AO_0041" audioObjectName="R123_4a">
  <audioObjectIDRef>AO_0002</audioObjectIDRef>
  <audioObjectIDRef>AO_0004</audioObjectIDRef>
</audioObject>

<audioObject audioObjectID="AO_0002" audioObjectName="R123_Stereo">
  <audioPackFormatIDRef>AP_00010002</audioPackFormatIDRef>
  <audioTrackUIDRef>ATU_00000001</audioTrackUIDRef>
  <audioTrackUIDRef>ATU_00000002</audioTrackUIDRef>
</audioObject>

<audioObject audioObjectID="AO_0004" audioObjectName="R123_5.1coded">
  <audioPackFormatIDRef>AP_00010003</audioPackFormatIDRef>
  <audioTrackUIDRef>ATU_00000003</audioTrackUIDRef>
  <audioTrackUIDRef>ATU_00000004</audioTrackUIDRef>
</audioObject>

<!-- ############ -->
<!—- AUDIO TRACK UIDs -->
<!-- ############ -->

<audioTrackUID UID="ATU_00000001">
  <audioMXFLookUp>
<packageUIDRef>urn:smpte:umid:060a2b34.01010105.01010f20.13000000.540bca53.41434f05.8ce5f4e
3.5b72c985</packageUIDRef>
    <trackIDRef>MXFTRACK_3</trackIDRef>
    <channelIDRef>MXFCHAN_1</channelIDRef>
  </audioMXFLookUp>
  <audioTrackFormatIDRef>AT_00010001_01</audioTrackFormatIDRef>
  <audioPackFormatIDRef>AP_00010002</audioPackFormatIDRef>
</audioTrackUID>

<audioTrackUID UID="ATU_00000002">
  <audioMXFLookUp>
<packageUIDRef>urn:smpte:umid:060a2b34.01010105.01010f20.13000000.540bca53.41434f05.8ce5f4e
3.5b72c985</packageUIDRef>
    <trackIDRef>MXFTRACK_3</trackIDRef>
    <channelIDRef>MXFCHAN_2</channelIDRef>
  </audioMXFLookUp>
  <audioTrackFormatIDRef>AT_00010002_01</audioTrackFormatIDRef>
  <audioPackFormatIDRef>AP_00010002</audioPackFormatIDRef>
</audioTrackUID>

<audioTrackUID UID="ATU_00000003">
  <audioMXFLookUp>

<packageUIDRef>urn:smpte:umid:060a2b34.01010105.01010f20.13000000.540bca53.41434f05.8ce5f4e
3.5b72c985</packageUIDRef>
    <trackIDRef>MXFTRACK_3</trackIDRef>
    <channelIDRef>MXFCHAN_1</channelIDRef>
  </audioMXFLookUp>
  <audioTrackFormatIDRef>AT_00020001_01</audioTrackFormatIDRef>
</audioTrackUID>

<audioTrackUID UID="ATU_00000004">
```

```
  <audioMXFLookUp>
<packageUIDRef>urn:smpte:umid:060a2b34.01010105.01010f20.13000000.540bca53.41434f05.8ce5f4e
3.5b72c985</packageUIDRef>
    <trackIDRef>MXFTRACK_3</trackIDRef>
    <channelIDRef>MXFCHAN_1</channelIDRef>
  </audioMXFLookUp>
  <audioTrackFormatIDRef>AT_00020001_02</audioTrackFormatIDRef>
</audioTrackUID>
```