

Immersive Audio Bitstream Specification



Page 1 of 54

Table of Contents

	Page
1	Scope
2	Conformance Notation
3	Normative References
4	Terms and Definitions
4.1	Bed (also, Immersive Audio Bed)
4.2	IABitstream frame
4.3	IAFrame
4.4	IAFrame Rate
4.5	Target Environment
5	Coding of Numbers and Data Types
5.1	Encoding of data fields
5.2	Plex encoding
5.3	UseCase encoding
5.4	Relative distance coding
5.5	Amplitude Gain
6	Bitstream Container and Syntax Description - Informative
7	Bitstream Segment Specification
8	IABitstream Field Description
8.1	IABitstream data fields
9	Bitstream IAFrame Specification
9.1	IAFrame Syntax
9.2	BedDefinition Syntax
9.3	BedRemap Syntax
9.4	ObjectDefinition Syntax
9.5	ObjectZoneDefinition19 Syntax
9.6	AudioDataDLC Syntax
9.7	AudioDataPCM Syntax
9.8	AuthoringToolInfo Element
9.9	UserData Element
10	IAFrame Data Fields
10.1	IAElement Data Fields
10.2	IAFrame data fields
10.3	BedDefinition Data Fields
10.4	BedRemap Data Fields
10.5	ObjectDefinition Data Fields

10.6	ObjectZoneDefinition19 Data Fields	32
10.7	AudioDataDLC Data Fields	33
10.8	AudioDataPCM Data Fields	36
10.9	AuthoringToolInfo Element	36
10.10	UserData Element	36
11	Bitstream Conventions	36
11.1	Position	36
11.2	Bitstream Constants	37
Annex A	Example Bitstreams – Informative	38
A.1	Example Bitstream Organization for Two Simultaneous Beds	38
A.2	Example Bitstream Organization for Replacement Beds	39
A.3	Example Bitstream Organization for Bed Remapping	40
A.4	Example Bitstream Organization for Simultaneous Bed and Object Rendering	41
A.5	Example Bitstream Organization for Replacement Objects	42
Annex B	Sample Rate Scalable Lossless Audio Coding (AudioDataDLC)	43
B.1	Sample rate scalable lossless coding, Overview	43
B.2	Encoder	43
B.3	Overview (Informative)	43
B.4	Decoder	45
B.5	Overview	45
B.6	Decoding the AudioDataDLC Audio Data	46
B.7	Converting the Lattice Prediction Coefficients to Direct Form Coefficients	47
B.8	Applying the Predictors	48
B.9	Up-sampling and Interpolation of the 48 kHz Base Layer	49
B.10	Adding Base and Extension Layer and Perform Final LSB Shifting	50
B.11	Minimal DLC Encoder (Informative)	52
Annex C	Constraints	54
C.1	General Bitstream Constraints	54

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 25CSS.

Intellectual Property

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.

Introduction

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

This document has two main sections to reflect the structure of the Immersive Audio Bitstream. The structure is defined to have two levels of wrapping. At the top level, there are two defined segments as defined in Section 7: the preamble segment and the IAFrame segment. This wrapping is documented in order to comply with existing practice, but does not add value to the bitstream. The second level of wrapping is within the IAFrame and is described starting in Section 9. All immersive audio data is carried in the IAFrame.

The Immersive Audio Bitstream can carry audio channels (bed channels) that are destined for particular Loudspeakers or Loudspeaker arrays within the auditorium. The bitstream can also carry Audio Objects that are "rendered" to particular Loudspeakers or combinations of Loudspeakers based on positional data in the bitstream and auditorium-specific information about Loudspeakers and their locations. Bitstream immersive audio data is broken into IAFrames, each corresponding to an image edit unit. For example, if the image edit rate is 24 edit units per second, an IAFrame contains the audio to be played during that 1/24 second period of time. An IAFrame contains a number of AudioData elements (AudioDataPCM or AudioDataDLC), each containing a single audio waveform encoded as PCM or a losslessly compressed version of PCM.

Bed definition elements tell the renderer which particular audio element should be routed to which Loudspeaker or array of Loudspeakers. Bed definition elements "point to" AudioData elements using the AudioDataID and Loudspeaker channels using the ChannelID. Bed remap elements are sub elements of Bed definition elements. Bed remap elements define how a Bed mixed for one Loudspeaker configuration can be played in another.

Object elements tell the renderer which particular AudioData element should be "panned" to a particular location in the auditorium. As with a Bed channel, an object element "points to" AudioData elements and pans the audio as per the panning instructions included in the object element. Because the object may be

moving during the IAFRame, the object element is broken into “sub blocks” to increase the update rate of position data.

1 Scope

The normative portions of this document define a coded representation (bitstream) of an audio program. Information is included to describe typical uses of the format. The bitstream carries audio essence and metadata necessary to reproduce a complete audio program.

2 Conformance Notation

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

All text in this document is, by default, normative, except: the Introduction, any section 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; followed by formal languages; then figures; and then any other language forms.

3 Normative References

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

SMPTE ST 2098-1:2018 Immersive Audio Metadata

SMPTE ST 298:2009, Universal Labels for Unique Identification of Digital Data

SMPTE ST 428-12:2013, D-Cinema Distribution Master - Common Audio Channels and Soundfield Groups

SMPTE ST 2098-5:2018, D-Cinema Immersive Audio Channels and Soundfield Groups

IETF RFC 2396, Uniform Resource Identifiers (URI): Generic Syntax

ITU-R BS.2051-2, Advanced sound system for programme production

4 Terms and Definitions

For the purposes of this document, the following terms and definitions apply. See Normative References for definitions of other terms used in this document.

4.1 Bed (also, Immersive Audio Bed)

Soundfield Group, such as a 5.1, 7.1 or 9.1, that serves as the foundation of the immersive soundtrack mix

Note: A Bed is typically present for the duration of the program. Audio Objects may be added to the immersive soundtrack mix to augment the Bed. The Loudspeaker assignments for a Bed are typically static and do not change during the program.

4.2 IABitstream frame

Smallest editable unit of the Immersive Audio Bitstream

4.3 IAFrame

Portion of an IABitstream frame that contains all audio essence and metadata necessary to reconstruct one frame of audio.

4.4 IAFrame Rate

Number of IAFrames per second

Note: The IAFrame Rate normally matches the edit rate of an associated picture track.

4.5 Target Environment

Specific set of conditions that is present in the playback environment

5 Coding of Numbers and Data Types

5.1 Encoding of data fields

Unless otherwise specified, bit stream data fields that are larger than 1 bit shall be encoded as follows: most-significant-bit-first for fields that represent numeric data (e.g. PreambleLength); or left-bit-first for

fields that represent non-numeric data (e.g. UseCaseCode). When the bitstream is represented as a bytestream, the first bit of the bitstream shall correspond to the most significant bit of the first byte of the bytestream. Consecutive bits of the bitstream shall correspond in sequence to the remaining bits of the first byte and then in the same pattern to the bits of consecutive bytes of the bytestream.

5.2 Plex encoding

Plex(n) defines a field size of n bits that may be expanded if it is insufficient to hold the desired symbol. A data field of all ones is an escape code. If the desired symbol is not the escape code and fits within the allocated field, the symbol shall be placed therein, with leading zeros (as needed) to fill the field. Otherwise, an escape code is used to signal that a replacement symbol follows in the bitstream. The field size for the replacement symbol is twice as long as the field it replaces. This process continues, with additional escape codes and doubling of field size, until a field with size sufficient to carry the desired symbol is provided. The desired symbol shall be placed therein, with leading zeros (as needed) to fill the field. Prior to Plex coding, any leading zeros of a symbol (e.g., an unsigned integer) shall be dropped in order to fit the symbol into a smaller container. During decoding, leading zeros shall be prepended as needed to reproduce the full symbol. Plex coded symbols shall be encoded in the smallest container possible for the value to be encoded. For example, 0x1234 is to be Plex(8) encoded as 0xFF1234 instead of 0xFFFFFFF00001234. In this document, symbols to be Plex encoded shall have a value less than or equal to 0xFFFFFFFFE.

Informative Note: Example Plex encoding.

Encode the 32-bit number 0x12345678 as Plex(4).

F – Number does not fit in 4 bits. Put down escape code and double field size to 8 bits.

FF – Number does not fit in 8 bits. Put down escape code and double field size to 16 bits.

FFFF – Number does not fit in 16 bits. Put down escape code and double field size to 32 bits.

12345678 – Number fits in 32 bits.

Full coding is F FF FFFF 12345678.

Encode the 32-bit number 0x12345678 as Plex(8).

FF – Number does not fit in 8 bits. Put down escape code and double field size to 16 bits.

FFFF – Number does not fit in 16 bits. Put down escape code and double field size to 32 bits.

12345678 – Number fits in 32 bits.

Full coding is FF FFFF 12345678

Encode the 8-bit number 0xFF as Plex(8)

FF – The number fits in 8 bits, but FF is reserved as an escape. Put down escape code and double field size to 16 bits

00FF – Number fits in 16 bits.

Full coding is FF 00FF.

5.3 UseCase encoding

A UseCase code is an 8-bit value that indicates the Target Environment for which a bitstream element is intended. The defined UseCase codes are shown in Table 1 and are associated with Soundfield Configurations as defined in Table 1. Note: Decoders can compare the UseCase code associated with an element to configuration information, such as the Loudspeakers used for channel support, to determine whether that element should be used.

Table 1 - UseCase Codes

UseCase code	General Description	SMPTE ST 428-12:2013 (Annex C) Reference	SMPTE ST 2098-5:2018 (Annex C) Reference	ITU-R BS.2051-2 Reference
0x01	Intended for use with a specific Soundfield Configuration	5.1		
0x02	Intended for use with a specific Soundfield Configuration	7.1DS		
0x03	Intended for use with a specific Soundfield Configuration	7.1SDS		
0x04	Intended for use with a specific Soundfield Configuration		11.1HT	
0x05	Intended for use with a specific Soundfield Configuration		13.1HT	
0x06	Intended for use with a specific Soundfield Configuration		9.1OH	
0x07-0x2F	Reserved for D-Cinema			
0x30	Intended for use with a specific Soundfield Configuration			Speaker Layout A
0x31	Intended for use with a specific Soundfield Configuration			Speaker Layout D
0x32	Intended for use with a specific Soundfield Configuration			Speaker Layout J
0xFF	Always Use			
All other values	Reserved			

5.4 Relative distance coding

Distance Coding

Throughout the bitstream, distance metadata on or within the unit cube shall be coded as an n -bit unsigned integer (D_n) that maps linearly into the range $[0,1]$, but the mapping depends on the axis. For the X and Y axis, the DistanceXY formula shall be used. For the Z axis, the DistanceZ formula shall be used. See Section 11.1 for an explanation of the unit cube and the associated axes.

$$\text{DistanceXY} = D_n / 2^{(n-1)} - (2^{(n-1)} - 1) / 2^{(n-1)}, \quad 2^{n-1} - 1 \leq D_n \leq (2^n) - 1$$

$$\text{DistanceZ} = D_n / (2^n - 1), \quad 0 \leq D_n \leq (2^n) - 1$$

5.5 Amplitude Gain

Gains, except zone gains, shall be expressed as a 10-bit unsigned integer. If the 10-bit value (G) is 0x3ff, samples shall be multiplied by zero (linear gain is zero or negative infinity dB). Otherwise, samples shall be multiplied by $2^{-G/64}$. For example, if the 10-bit encoded value G is 0, the linear gain is $2^0 = 1$. Samples are multiplied by 1. If $G = 64$ (0x40), samples are multiplied by $2^{-1} = 1/2$, which corresponds to a gain of -6 dB.

6 Bitstream Container and Syntax Description - Informative

The Immersive Audio Bitstream is composed of a number of sequential frames, where each frame is an IABistream frame and contains a Preamble segment and an IAFrame segment. The Preamble segment and tag-length-value wrapping of the IAFrame are documented to allow existing files to be compliant with this specification and do not provide any other function. The bulk of this document describes the IAFrame.

The IAFrame uses a hierarchical container format composed of “elements” similar in concept to “chunks” in the RIFF¹ format. Each element consists of a header and payload. The header consists of a unique identifier (ElementID), and the ElementSize. The ElementID specifies how the data within the element is interpreted. ElementSize indicates the size in bytes of the following payload. The bits to code ElementID and ElementSize are not included in the calculation of ElementSize. The element payload may contain sub elements. The ElementSize includes the size of all sub elements contained within the payload. Sub elements contain additional description data related to the parent element.

The bitstream specification in Sections 7 and 9 is documented using pseudocode. The pseudocode describes the order of arrival of symbols within the bitstream. The pseudocode is roughly based on C language syntax, but simplified for ease of reading. Symbols carried in the bitstream are designated by bold face font.

The length of each symbol (symbol length) is indicated in bits to the right of the symbol name. In some cases, the symbol is coded using a method called Plex coding to allow for variable length symbols. See Section 5.2 for details.

¹ Resource Interchange File Format

7 Bitstream Segment Specification

The Immersive Audio bitstream shall be a sequence of IABitstream frames. Each IABitstream frame consists of two tag-length-value wrapped segments: a preamble segment and an IAFrame segment as shown in Table 2. Successive frames shall correspond to adjoining, non-overlapping audio segments.

Table 2 - IABitstream Syntax

IABitstream Syntax	symbol length
IAB IABitstream	
{	
while(true)	
{	
PreambleTag	8
PreambleLength.....	32
PreambleValue.....	variable
IAFrameTag.....	8
IAFrameLength.....	32
IAFrame	variable
}	
} /* end of IABitstream */	

8 IABitstream Field Description

The following sections define the bit fields referenced by name in the previous section. Data fields are defined in the following sections, and each field name is followed by the number of bits used by the field.

8.1 IABitstream data fields

8.1.1 PreambleTag – 8 bits

The PreambleTag value shall be equal to 0x01.

8.1.2 PreambleLength – 32 bits

The PreambleLength field shall indicate the size, in bytes, of the PreambleValue field. The PreambleLength field may have a value of zero.

8.1.3 PreambleValue – Variable

The PreambleValue field shall be comprised of an integer number of payload bytes. Specification of the content of the payload is outside the scope of this specification.

8.1.4 IAFrameTag – 8 bits

The IAFrameTag value shall be 0x02.

8.1.5 IAFrameLength – 32 bits

The IAFrameLength field shall indicate the size, in bytes, of the IAFrame.

8.1.6 IAFRAME – Variable

The IAFRAME is a data structure described in Section 9.

9 Bitstream IAFRAME Specification

Each IAFRAME contains all the information (metadata and essence) necessary to describe the entire audio program for one time segment. The duration of each coded time segment shall correspond to one of the supported frame rates as listed in Table 17. The audio frames shall be aligned with the audio program edit units. In this document, a Frame shall mean an IAFRAME element (see Section 9.1). The IAFRAME is an instance of an IAELEMENT syntax.

The IAFRAME uses a hierarchical container format composed of “elements.” The IAELEMENT syntax in Table 3 describes the header structure that begins all Immersive Audio elements.

Table 3 - IAELEMENT Syntax

IAELEMENT Syntax	symbol length
<pre> IAELEMENT { ELEMENTID.....Plex(8) ELEMENTSIZE.....Plex(8) switch(ELEMENTID) { case(IA_FRAME) IAFRAME break case(BED_DEFINITION) BedDefinition break case(BED_REMAP) BedRemap break case(OBJECT_DEFINITION) ObjectDefinition break case (OBJECT_ZONE_DEFINITION19) ObjectZoneDefinition19 break case (AUTHORING_TOOL_INFO) AuthoringToolInfo break case (USER_DATA) UserData break case(AUDIO_DATA_DLC) AudioDataDLC break case(AUDIO_DATA_PCM) AudioDataPCM break default UnknownData.....ElementSize * 8 } </pre>	

IAElement Syntax	symbol length
} /* end of IAElement */	

Table 4 specifies which Sub-Elements an IAElement can carry.

Table 4 - Allowed sub-elements.

Element Name	Allowed Sub-Elements
IAFrame	BedDefinition ObjectDefinition AudioDataDLC AudioDataPCM AuthoringToolInfo UserData
BedDefinition	BedDefinition BedRemap
BedRemap	None
ObjectDefinition	ObjectDefinition ObjectZoneDefinition19
AudioDataDLC	None
AudioDataPCM	None
AuthoringToolInfo	None
UserData	None
ObjectZoneDefinition19	None

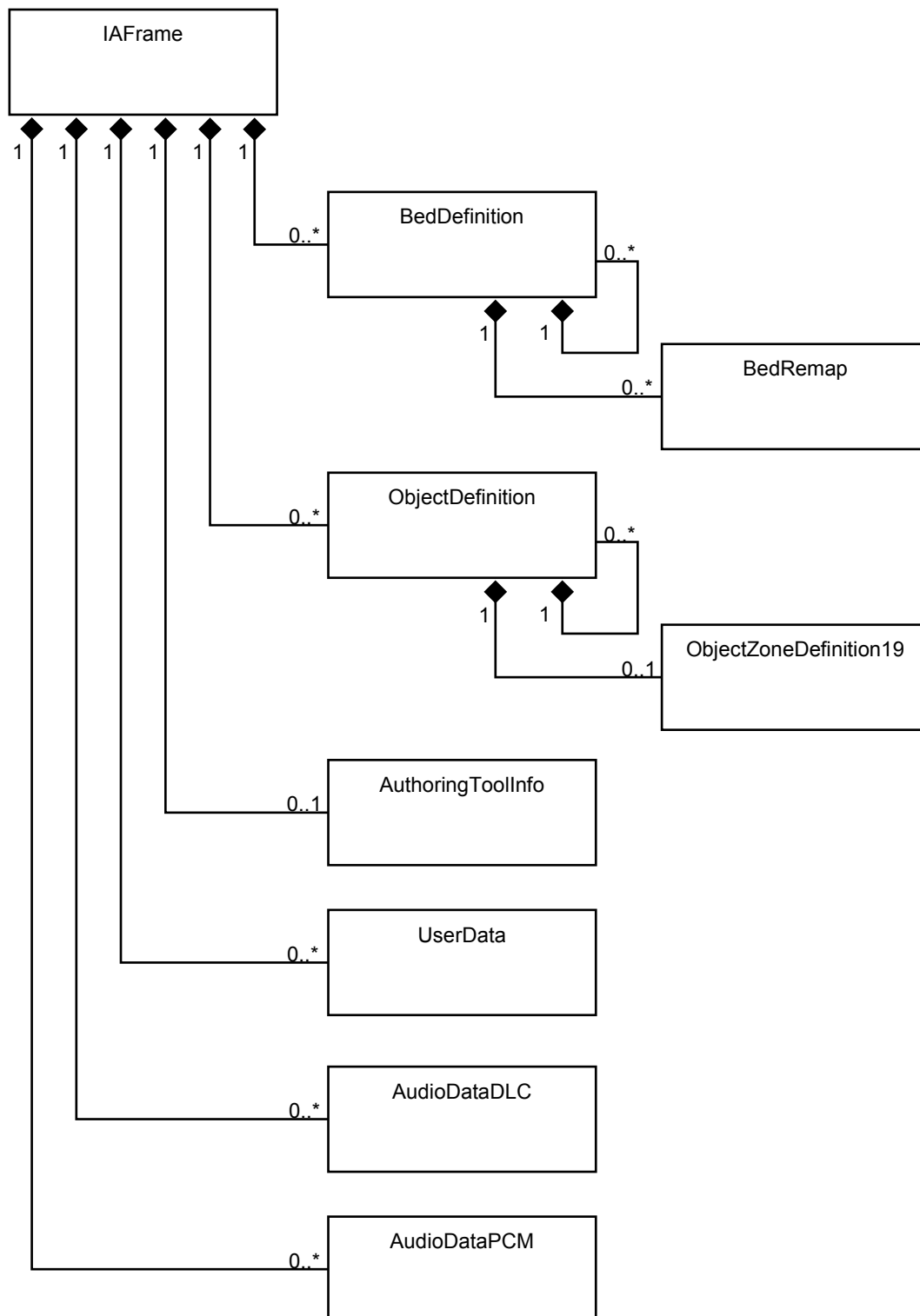


Figure 1 – UML class diagram showing possible element parent-child relationships and cardinalities

Figure 1 illustrates the possible element parent-child relationships and cardinalities. A connecting line with a diamond shape at one end indicates a composition aggregation meaning that an Element type on the side of the diamond shape is composed of Element types on the other side of the connecting line. The number of Elements that the Element at the side with the diamond shape is composed of is given by the cardinality at the other end of the connecting line. A cardinality of “0..*” is interpreted as “zero or more”. For example, the composition of an IAFRAME Element includes zero or more AudioDataDLC Elements. A reflexive composition is indicated by a connecting line starting and ending at the same Element type. For example, an ObjectDefinition Element can be composed of zero or more further ObjectDefinition Elements.

9.1 IAFRAME Syntax

The IAFRAME element contains all audio and metadata elements required to decode one frame of audio. The IAFRAME element shall have the structure shown in Table 5. The following elements shall be supported as children of an IAFRAME: BedDefinition, ObjectDefinition, AudioDataDLC, AudioDataPCM, AuthoringToolInfo, and UserData. Other child elements shall be ignored.

Table 5 - IAFRAME Syntax

IAFRAME Syntax	symbol length
IAFRAME	
{	
Version	8
SampleRate	2
BitDepth	2
FrameRate	4
MaxRendered	Plex(8)
AlignBits /* extra bits to get to byte alignment relative to the start of the element */	
SubElementCount	Plex(8)
for(n = 0; n < SubElementCount; n ++){	
IAElement	
}	
} /* end of IAFRAME */	

9.2 BedDefinition Syntax

The BedDefinition element contains metadata and pointers to audio essence to generate one frame of one audio Bed. The BedDefinition element shall have the structure shown in Table 6. The following elements shall be supported as children of a BedDefinition element: BedDefinition and BedRemap. Other child elements shall be ignored.

Table 6 - BedDefinition Syntax

BedDefinition Syntax	symbol length
BedDefinition	
{	
MetaID	Plex(8)
ConditionalBed	1
if(ConditionalBed == 1){	

BedDefinition Syntax	symbol length
BedUseCase	8
}	
ChannelCount	Plex(4)
for(n = 0; n < ChannelCount; n ++){	
ChannelID [n]	Plex(4)
AudioDataID [n]	Plex(8)
ChannelGainPrefix	2
if(ChannelGainPrefix > 1){	
ChannelGain [n]	10
}	
ChannelDecorInfoExists	1
if(ChannelDecorInfoExists == 1){	
Reserved (set to 0)	4
ChannelDecorCoefPrefix	2
If(ChannelDecorCoefPrefix>1) {	
ChannelDecorCoef [n]	8
}	
}	
}	
Reserved (set to 0x180)	10
/* reads extra bits to get to byte alignment relative to the start of the element*/	
AlignBits (set to 0)	VARIABLE
AudioDescription	8
if(AudioDescription & 0x80){	
n = -1	
do {	
/* NULL-terminated, strict ASCII Text */	
n++	
AudioDescriptionText [n]	8
}while(AudioDescriptionText[n] != 0x00)	
}	
SubElementCount	Plex(8)
for(n = 0; n < SubElementCount; n ++){	
IAElement	
}	
} /* end of BedDefinition */	

9.3 BedRemap Syntax

BedRemap is an optional element that describes how to convert an audio Bed from the distribution configuration to a different playback configuration. For example, BedRemap can provide information to downmix a 9.1 Bed to a 5.1 Bed for playback in a 5.1 cinema. The BedRemap element shall have the structure shown in Table 7. The BedRemap element shall only be used as a child-element of a BedDefinition element.

On a sub block by sub block basis, BedRemap provides mixing coefficients to distribute each channel of the parent BedDefinition (bed channel) into zero or more of the channels supported by the playback system (playback channel). For example, the BedRemap could specify that the Center Bed channel be distributed with the following coefficients for a stereo playback system: CtoL_gain = 0.7, CtoR_gain = 0.7.

There is additional processing required to convert channel signals to Loudspeaker feeds including the application of the decorrelation and ChannelGains specified in the BedDefinition element. Note: The iChan in BedRemap corresponds to n in the ChannelCount 'for loop' in the BedDefinition element.

Table 7 - BedRemap Syntax

BedRemap Syntax	Symbol length
<pre> BedRemap { MetaID Plex(8) RemapUseCase 8 SourceChannels Plex(4) DestinationChannels Plex(4) /* NumRemapSubBlocks is derived from the Sample Rate and Frame Rate and is the same as NumPanSubBlocks. See Table 23. */ for(subBlk = 0; subBlk < NumRemapSubBlocks; subBlk++){ if(subBlk == 0){ RemapInfoExists = 1 } else{ RemapInfoExists 1 } if(RemapInfoExists == 1){ for(oChan = 0; oChan < DestinationChannels; oChan++){ /* Destination channel position. See Table 19 */ DestinationChannelID[subBlk][oChan] Plex(4) for(iChan = 0; iChan < SourceChannels; iChan++){ RemapGainPrefix 2 if(RemapGainPrefix > 1) { RemapGain[subBlk][oChan][iChan] 10 } } } } } /* reads extra bits to get to byte alignment relative to the start of the element */ AlignBits(set to 0) VARIABLE Reserved(set to 0) Plex(8) } } /* end of BedRemap*/ </pre>	

9.4 ObjectDefinition Syntax

ObjectDefinition contains metadata, most importantly position, and pointers to audio essence to generate one frame of one Audio Object. The ObjectDefinition element shall have the structure shown in Table 8. The following elements shall be supported as child elements of the ObjectDefinition element: ObjectDefinition and ObjectZoneDefinition19. Other child elements shall be ignored.

Table 8 - ObjectDefinition Syntax

ObjectDefinition Syntax	symbol length
ObjectDefinition	
{	
MetaID Plex(8)	
AudioDataID Plex(8)	
ConditionalObject 1	
if(ConditionalObject == 1){	
Reserved (set to 1) 1	
ObjectUseCase 8	
}	
Reserved (set to 0).....1	
/* NumPanSubBlocks is dependent on frame rate. See Table 23 */	
for(sb = 0; sb < NumPanSubBlocks; sb++){	
if(sb == 0){	
PanInfoExists = 1	
}	
else{	
PanInfoExists 1	
}	
if(PanInfoExists == 1){	
ObjectGainPrefix 2	
if(ObjectGainPrefix > 1) {	
ObjectGain [sb].....10	
}	
Reserved (set to 0b001) 3	
ObjectPosX [sb] 16	
ObjectPosY [sb] 16	
ObjectPosZ [sb] 16	
ObjectSnap [sb] 1	
if(ObjectSnap[sb] == 1){	
ObjectSnapTolExists 1	
if(ObjectSnapTolExists== 1){	
ObjectSnapTolerance [sb] 12	
}	
Res2 [sb] (set to 0).....1	
}	
ObjectZoneControl 1	
if(ObjectZoneControl == 1){	

ObjectDefinition Syntax	symbol length
<pre> /* For each zone listed in Table 24 */ for(n = 0; n < 9; n ++){ ZoneGainPrefix 2 if(ZoneGainPrefix > 0x1){ ZoneGain[sb][n] 10 } } ObjectSpreadMode 2 if(ObjectSpreadMode == OBJECT_SPREAD_LOWREZ){ ObjectSpread[sb] 8 } else if(ObjectSpreadMode== OBJECT_SPREAD_1D){ ObjectSpread[sb] 12 } else if(ObjectSpreadMode == OBJECT_SPREAD_3D){ ObjectSpreadX[sb] 12 ObjectSpreadY[sb] 12 ObjectSpreadZ[sb] 12 } Reserved (set to 0) 4 ObjectDecorCoefPrefix 2 if(ObjectDecorCoefPrefix > 1){ ObjectDecorCoef[sb] 8 } } /* end if(PanInfoExists) */ /* reads extra bits to get to byte alignment relative to the start of the element */ AlignBits(set to 0) VARIABLE AudioDescription 8 if(AudioDescription & 0x80){ n = -1 do { /* NULL-terminated, strict ASCII Text */ n++ AudioDescriptionText[n] 8 }while(AudioDescriptionText[n] != 0x00) } SubElementCount Plex(8) for(n = 0; n < SubElementCount; n ++){ IAElement } } /* end of ObjectDefinition*/ </pre>	

9.5 ObjectZoneDefinition19 Syntax

ObjectZoneDefinition19 is an optional sub-element of an ObjectDefinition element. It describes an alternative zone definition that is associated with a different set of zones. An ObjectZoneDefinition19