SMPTE RDD 29:2019
Revision of RDD 29:2014

SMPTE REGISTERED DISCLOSURE DOCUMENT

Dolby Atmos® Bitstream Specification

The attached document is a Registered Disclosure Document prepared by the proponent identified below. It has been examined by the appropriate SMPTE Technology Committee and is believed to contain adequate information to satisfy the objectives defined in the Scope, and to be technically consistent.

This document is NOT a Standard, Recommended Practice or Engineering Guideline, and does NOT imply a finding or representation of the Society.

Errors in this document should be reported to the proponent identified below, with a copy to eng@smpte.org.

This document is intended to allow the interpretation of Dolby Atmos Bitstream files. It is not intended to support the development of hardware or software applications that create or process these files. Creation and processing of such files is reserved to individuals and organizations that have entered into agreements with the proponent identified below for this purpose. Use of this document to produce or process Dolby Atmos files using non-Dolby tools would potentially cause user confusion, diminished sound quality as experienced by content consumers, and damage to the reputation of the Dolby Atmos brand and to Dolby Laboratories itself.

All other inquiries in respect of this document, including inquiries as to intellectual property requirements that may be attached to use of the disclosed technology, should be addressed to the proponent identified below.

Proponent contact information:

Andrew Poulain
Dolby Laboratories Inc.
1275 Market St.
San Francisco, CA 94103

Email andrew.poulain@dolby.com

Approved November 20, 2018

Table of Contents

Page

Introduction

Dolby Atmos® is an advanced cinema sound format comprising an audio essence and metadata stream played through specialized renderers in the cinema.

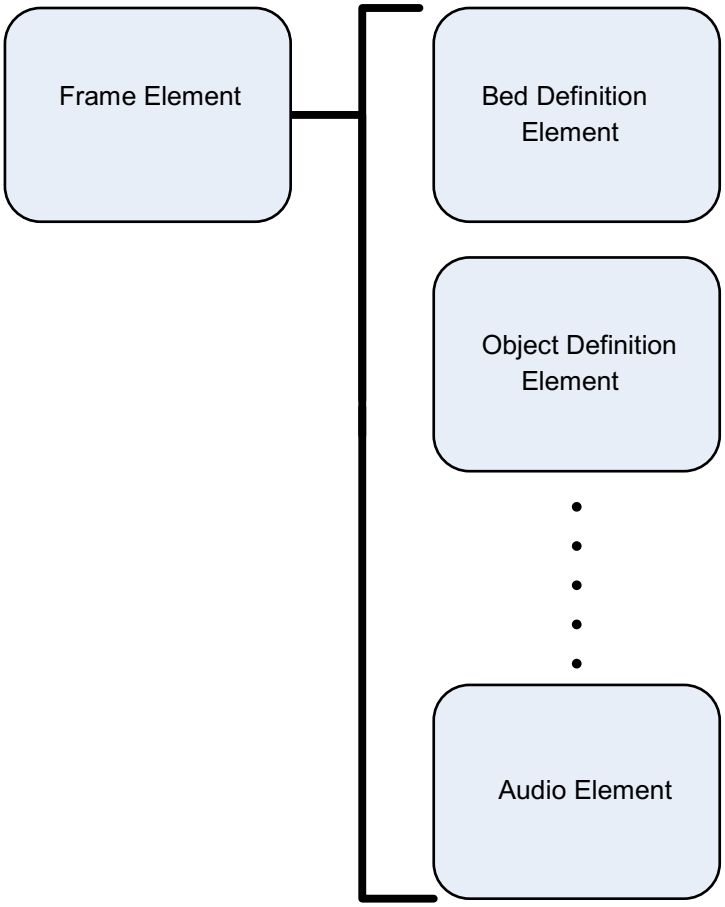
1 Scope

This document defines the syntax of a frame-based Dolby Atmos bit stream. The bit stream carries audio essence and metadata necessary to reproduce a complete audio program.

2 Bitstream Organization

The audio program is segmented into Frames, with Frames transmitted 24, 25, 30, 48, 50, 60, 96, 100, or 120 times per second. The audio Frames are aligned with the program edit units. In most cases the edit units, and picture and audio frames are of the same duration and are time aligned. Support for frame rates above 120 Hz is not defined.

All audio data is encapsulated into "elements," similar in concept to "chunks" in the RIFF[1] format. Each element begins with a unique identifier called **ElementID**. The second field, **ElementSize**, indicates the size in bytes of the entire element, not including the **ElementID** and **ElementSize**. Elements can contain sub-Elements. The **ElementSize** includes the size of all sub elements. Sub-Elements contain additional description data related to the parent Element. At the top level, the entire audio frame is contained in a single *ATMOSFrame* element. All audio essence and metadata elements for a given frame are contained as sub Elements of the ATMOSFrame Element as shown below.



[1] Resource Interchange File Format

Currently there are 4 element types specified in the Dolby Atmos bit stream; *ATMOSFrame*, *BedDefinition*, *ObjectDefinition*, and *AudioDataDLC*. The purpose of each element is described in the following sections.

2.1 ATMOSFrame Element

The *ATMOSFrame* element contains all the information that is common to the entire Dolby Atmos frame. Specifically, the *ATMOSFrame* contains the Dolby Atmos version, audio sample rate, the audio bit depth, the audio frame rate, and the maximum number of rendered audio assets. All raw audio assets and metadata must be sub elements of the *ATMOSFrame* element.

2.2 BedDefinition Element

A Dolby Atmos bed is a collection of audio channels. An audio channel is an audio stream that is intended to be played back with a nominal location (e.g. “Left” channel) or function (e.g. LFE). The *BedDefinition* element contains a list of the audio assets and the associated channel names.

2.3 ObjectDefinition Element

The Dolby Atmos system allows audio assets to be panned to any location independent of the physical or nominal loudspeaker configuration; these panned audio assets are called objects. The *ObjectDefinition* element provides all the information to pan an audio object. Each *ObjectDefinition* element updates the position of a single audio object at approximately 20-ms time intervals.

The Dolby Atmos presentation can have a large number of audio objects that will be independently rendered to the appropriate locations. To achieve this, the Dolby Atmos bit stream will contain multiple *ObjectDefinition* elements that must be direct sub-elements of the *ATMOSFrame* element and must have a unique **MetaID**.

2.4 AudioDataDLC Element

Each *AudioDataDLC* element contains the audio assets for one track of audio, channel or object. Every audio track is losslessly compressed and exists for the duration of the program. The *AudioDataDLC* element supports sample rates of 48 kHz or 96 kHz with 24-bit resolution. All *AudioDataDLC* elements must be direct sub-elements of the *ATMOSFrame* element and must have a unique **AudioDataID**. Note: audio object tracks are typically sparse; most audio events conveyed by objects have limited time extent, with digital zero signal between events. The AudioDataDLC element can efficiently indicate periods of silence to dramatically decrease the audio payload.

3 Bitstream Conventions

3.1 Position

Axes and Origin
Many of the metadata elements contained in the bitstream specify a relative position or size. In most cases position is described relative to the playback environment using a unit cube to describe the room boundaries. The origin is taken to be the front left corner of the room. Position is then described using Euclidian (x,y,z) coordinates, assigned as follows:

x: lateral, or left/right position
x=0 corresponds to left wall; x=1 corresponds to right wall.

y: longitude, or front/back position
y=0 corresponds to front wall; y=1 corresponds to back wall.

z: elevation, or up/down position

z=0 corresponds to a plane aligned with the screen, side and rear loudspeakers;
z=1 corresponds to the ceiling.

For example,

(0, 0, 0) -> front left corner, 0 elevation (left screen speaker),

(1, 0, 0) -> front right corner, 0 elevation (right screen speaker), and

(0.5, 0.5, 1) -> middle of ceiling.

Metadata that describes position relative to the room uses the unit axes and origin described above; the location along each axis is coded using the distance coding method described below.

3.2 Relative distance coding

12 Bit Distance

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 10.1 for an explanation of the unit cube and the associated axes.

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

$$DistanceZ = D_n/(2^n – 1), \quad 0 <= D_n <= (2^n) – 1$$

3.3 Amplitude Gain

Many of the metadata elements contained in the bitstream specify a gain or scale factor to be applied to audio data. Throughout the bitstream, amplitude information is coded as a logarithmic value using a 10 bit mantissa, A10, to express values in the range [0, 1]. Thus the logarithmic code word, interpreted as an unsigned integer, can be mapped to a linear value as follows:

gain = 0, A10 = (2^10) - 1

gain = 2^(-A10/64), 0 <= A10 <= (2^10) – 2

This provides gain factors logarithmically spaced from 0 dB to -96 dB in steps of approximately 0.094 dB, plus a gain factor of zero (-infinity dB).

3.4 Plex Coding

Element size and some other fields are coded using Plex. Plex coding allows efficient coding of small numbers, with the capability to express arbitrarily large values. The ElementSize field is Plex(8) coded to efficiently indicate elements up to 254 bytes in length, and allow for arbitrarily large Element size.

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 0xFFFFFF00001234. In this document, symbols to be Plex encoded shall have a value less than or equal to 0xFFFFFFFE. Plex(8) coding is shown below.

Plex Syntax	Word Size
Plex(8)	
{	
PlexData	8
if(PlexData == 0xFF){	
PlexData	16
if(PlexData == 0xFFFFFF){	
PlexData	32
}	
}	
}	
return (PlexData)	
} /* end of Plex(8) */	

4 Bit Stream Syntax

The following pseudo code describes the order of arrival of information within the Dolby Atmos bit stream. The pseudo code is roughly based on C language syntax, but simplified for ease of reading. Field elements contained in the bit stream are designated by bold face font.

4.1 Syntax of ReadElement()

ReadElement syntax below describes the header structure that begins all the Dolby Atmos elements.

ReadElement Syntax	Word Size
ReadElement()	
{	
ElementID	Plex(8)
ElementSize	Plex(8)
switch(ElementID)	
{	
case(ATMOS_FRAME)	
ATMOSFrame()	
break	
case(BED_DEFINITION1)	
BedDefinition1()	
break	
case(OBJECT_DEFINITION1)	
ObjectDefinition1()	
break	
case(AUDIO_DATA_DLC)	
AudioDataDLC()	
break	
default	
UnknownData	ElementSize * 8
}	
} /* end of ReadElement */	

4.2 Syntax of ATMOSFrame( )

The ATMOSFrame element contains all audio and metadata elements required to decode one frame of audio.

ATMOSFrame Syntax	Word Size
ATMOSFrame()	
{	
ATMOSVersion	8
SampleRate	2
BitDepth	2
FrameRate	4
MaxRendered	Plex(8)
ByteAlign() /* extra bits to get to byte alignment relative to the start of the element */	
SubElementCount	Plex(8)
for(n = 0; n < SubElementCount; n ++){	
ReadElement()	
}	
} /* end of ATMOSFrame() */	

4.3 Syntax of BedDefinition1()

BedDefinition1 contains metadata and pointers to audio essence to generate one frame of one audio bed. An audio bed is a set of related audio channels in a defined configuration, e.g. 3.0, 5.1, or 9.1.

BedDefinition1 Syntax	Word Size
BedDefinition1()	
{	
MetaID	Plex(8)
Reserved (set to 0)	1
ChannelCount (set to 10)	Plex(4)
for(n = 0; n < ChannelCount; n ++){	
ChannelID[n]	Plex(4)
AudioDataID[n]	Plex(8)
Reserved (set to 0)	3
}	
Reserved (set to 0x180)	10
/* reads extra bits to get to byte alignment relative to the start of the element */	
AlignBits	VARIABLE
Reserved (set to 0x5)	8
Reserved (set to 0)	8
} /* end of BedDefinition1 */	

4.4 Syntax of ObjectDefinition1()

ObjectDefinition1 contains metadata, most importantly position, and pointers to audio essence to generate one frame of one audio object.

ObjectDefintion1 Syntax	Word Size
ObjectDefinition1()	
{	
MetaID	Plex(8)
AudioDataID	Plex(8)
Reserved (set to 0x7FE)	11
/* NumPanSubBlocks is dependent on audio sample rate and frame rate */	
for(sb = 0; sb < NumPanSubBlocks; sb ++){	
if(sb == 0){	
PanInfoExists = 1	
}	
else{	
PanInfoExists	1
}	
if(PanInfoExists == 1){	
Reserved (set to 0x1)	5
ObjectPosX[sb]	16
ObjectPosY[sb]	16
ObjectPosZ[sb]	16
ObjectSnap[sb]	1
if(ObjectSnap[sb] == 1){	
Reserved (set to 0)	2
}	
ObjectZoneControl[sb]	1
if(ObjectZoneControl[sb] == 1){	
for(n = 0; n < MAX_KNOWN_ZONES; n ++){	
ZoneGain	2
}	
}	
ObjectSpreadMode	2
if(ObjecSpreadMode == OBJECT_SPREAD_LOWREZ){	
ObjectSpread[sb]	8
}	
else if(ObjectSpreadMode== OBJECT_SPREAD_1D){	
ObjectSpread[sb]	12
}	
else{	
ObjectSpread[sb] = 0.0	
}	
Reserved (set to 0)	4
ObjectDecorCoefPrefix	2
if(ObjectDecorCoefPrefix > 1){	
ObjectDecorCoef[sb]	8
}	

ObjectDefintion1 Syntax	Word Size
} /* end if(PanInfoExists) */	
}	
/* reads extra bits to get to byte alignment relative to the start of the element */	
AlignBits	VARIABLE
AudioDescription	8
if(AudioDescription & 0x80){	
while(AudioDescription != 0x00){	
/* NULL Terminated ASCII Text Entry */	
AudioDescription	8
}	
}	
Reserved (set to 0)	8
} /* end of ObjectDefinition1 */	

4.5 Syntax of AudioDataDLC()

This element contains the audio essence. The audio tracks in this element are referenced by the audio bed and audio object elements.

AudioDataDLC Syntax	Word Size
AudioDataDLC()	
{	
AudioDataID	Plex(8)
DLCSize	16
DLCSampleRate	2
ShiftBits	5
/* Predictor information */	
NumPredRegions	2
for(n = 0; n < NumPredRegions; n ++) {	
RegionLength[n]	4
FIROrder[n]	5
for(m = 1; m <= FIROrder; m ++) {	
FIRPredictor[n][m]	10
}	
}	
/* Coded residual */	
for(n = 0; n < NumSubBlocks; n ++) {	
CodeType	1
if(CodeType == 0) {	
/* PCM Residual */	
BitDepth	5
for(l = 0; l < SubBlockSize; l ++) {	
Residual[n * SubBlockSize + l]	BitDepth
}	
}	
else {	

AudioDataDLC Syntax	Word Size
/*Rice/Golomb Residual */	
RiceRemBits	5
for(i= 0; i< SubBlockSize; i++) {	
quotient = 0	
UnaryBit	1
while(UnaryBit == 1){	
quotient ++	
UnaryBit	1
}	
Residual[n * SubBlockSize + i]	RiceRemBits
Residual[n * SubBlockSize + i] += quotient << RiceRemBits	
if(Residual96[n * SubBlockSize + i] != 0){	
sign	1
if(sign == 1){	
Residual[n * SubBlockSize + i] *= -1	
}	
}	
}	
}	
}	
/* 96kHz Residual Data */	
if(SampleRate == 0x1){	
/* Predictor information */	
NumPredRegions	2
for(n = 0; n < NumPredRegions; n ++) {	
RegionLength[n]	4
FIROrder[n]	5
for(m = 1; m <= FIROrder; m ++) {	
FIRPredictor[n][m]	10
}	
}	
/* Coded residual */	
for(n = 0; n < NumSubBlocks; n ++) {	
CodeType	1
if(CodeType == 0) {	
/* PCM Residual */	
BitDepth	5
for(l = 0; l < SubBlockSize; l ++) {	
Residual[n * SubBlockSize + l]	BitDepth
}	
}	
else {	
/*Rice/Golumn Residual */	
RiceCode	5
for(l = 0; l < SubBlockSize; l ++) {	
Residual[n * SubBlockSize + l]	VARIABLE
}	
}	
}	
}	

AudioDataDLC Syntax	Word Size
/* Each Element must keep track of the number of bits read */	
AlignBits	VARIABLE
} /* end of AudioDataDLC*/	

5 Bit Stream Field Description

5.1 ReadElement() Data Fields

5.1.1 ElementID – Plex(8)

Each Element block starts with an ElementID. The ElementID defines the type of element and the contents of the element. Depending on the ElementID the decoder will perform different tasks. Table 1 provides a list of ElementIDs. If the ElementID is not defined in the system, then the decoder shall skip the element.

Table 1 – Dolby Atmos Element IDs

ElementID Name	Value	Meaning
ATMOS_FRAME	0x08	Frame Header
BED_DEFINITION1	0x10	Bed Definition Type 1
RESERVED	0x20	Reserved
OBJECT_DEFINITION1	0x40	Object Definition Type 1
RESERVED	0x80	Reserved
RESERVED	0x100	Reserved
AUDIO_DATA_DLC	0x200	Audio Data (DLC encoded)

5.1.2 ElementSize – Plex(8)

ElementSize, indicates the size in bytes of the entire element, not including the ElementID and ElementSize.

For the Frame Element, the ElementSize is the entire audio frame (not including the ATMOSFrame ElementID and ElementSize) as all other elements are contained as sub elements.

5.2 ATMOSFrame() data fields

5.2.1 ATMOSVersion – 8 bits

The ATMOSVersion specifies the version of the Dolby Atmos Bit Stream. This field currently has the value of 0x1. **This document describes the protocol with ATMOSVersion = 1;**

5.2.2 SampleRate – 2 bits

The SampleRate code specifies the sampling rate of the audio data. All audio tracks, channels and objects, must have the same sampling rate. The SampleRate code has following definitions as shown in Table 2.

Table 2 – Sample Rate code

SampleRate Code	Meaning
0x0	48000 samples per second
0x1	96000 samples per second
0x2	RESERVED
0x3	RESERVED

5.2.3 BitDepth – 2 bits

The BiteDepth code specifies the bit depth of the object audio data. All audio tracks must have same bit depth. The BitDepth code has the following meanings as specified in Table 3. **Only 24-bits per audio sample are currently supported.**

Table 3 – Bit Depth Code

BitDepth Code	Meaning
0x0	RESERVED
0x1	24 bits per audio sample
0x2	RESERVED
0x3	RESERVED

5.2.4 FrameRate– 4 bits

The FrameRate code specifies the audio frame rate. The FrameRate code has the following meanings as specified by Table 4.

Table 4 – Frame Rate Code

FrameRate Code	Meaning
0x0	24 frames per second
0x1	25 frames per second
0x2	30 frames per second
0x3	48 frames per second
0x4	50 frames per second
0x5	60 frames per second
0x6	96 frames per second
0x7	100 frames per second
0x8	120 frames per second
0x9-0xF	RESERVED

The FrameRate code also controls the sample count (SampleCount) contained in each audio asset as specified by Table 5.

Table 5 – Sample Count versus Frame Rate Code and Sample Rate

FrameRate Code	Sample Count @ 48 kHz	Sample Count @ 96 kHz
0x0	2000	4000
0x1	1920	3840
0x2	1600	3200
0x3	1000	2000
0x4	960	1920
0x5	800	1600
0x6	500	1000
0x7	480	960
0x8	400	800
0x9-0xF	RESERVED	RESERVED

5.2.5 MaxRendered – Plex(8)

The MaxRendered code specifies the maximum audio assets that will be rendered during playback of the Dolby Atmos frame for theaters with the optimal target playback. For example, for a stream with 9.1 channel beds and 118 objects, the MaxRendered count would be set to 128.

5.2.6 SubElementCount – Plex(8)

The SubElementCount code is the number of elements contained in the current element.

5.3 BedDefinition1() Data Fields

5.3.1 MetaID – Plex(8)

MetaID is the unique ID that aids the system track metadata information between audio frames.

5.3.2 ChannelCount – Plex(4)

The channel count is the number of channels that make up the bed.

5.3.3 ChannelID – Plex(4)

The ChannelID code specifies the known channel locations. Table 6 provides a list of channelIDs and the associated loudspeaker name.

Table 6 – Channel IDs

ChannelID Code	Meaning
0x0	Left Screen Speaker
0x1	Left Center Screen Speaker
0x2	Center Screen Speaker
0x3	Right Center Screen Speaker
0x4	Right Screen Speaker
0x5	Left Side Surround Speaker (7.1)
0x6	Left Surround Speaker
0x7	Left Rear Surround Speaker (7.1)
0x8	Right Rear Surround Speaker (7.1)
0x9	Right Side Surround Speaker (7.1)
0xA	Right Surround Speaker
0xB	Left Top Surround Speaker (9.1)
0xC	Right Top Surround Speaker (9.1)

0xD	LFE Speaker
otherwise	Reserved

5.3.4 AudioDataID – Plex(8)

The AudioDataID code is a unique identifier to each of the raw mono audio assets carried in the bit stream. An AudioDataID of NULL (0) indicates no audio asset.

5.4 ObjectDefinition1() Data Fields

5.4.1 NumPanSubBlocks – Informative

The NumPanSubBlocks specifies the division of the frame into sub frames of approximately 5 ms, as specified by Table 7.

Table 7 – Number of Pan Sub Blocks and Sub Block Size versus Sample Rate and Frame Rate

Sample Rate	Frame Rate (sec^-1)	NumPanSubBlocks	PanSubBlockSize	Duration (ms)
48 kHz	24	8	250	5.2
48	25	8	240	5.0
48	30	8	200	4.2
48	48	4	250	5.2
48	50	4	240	5.0
48	60	4	200	4.2
48	96	2	250	5.2
48	100	2	240	5.0
48	120	2	200	4.2
96 kHz	24	8	500	5.2
96	25	8	480	5.0
96	30	8	400	4.2
96	48	4	500	5.2
96	50	4	480	5.0
96	60	4	400	4.2
96	96	2	500	5.2
96	100	2	480	5.0
96	120	2	400	4.2

5.4.2 PanInfoExists – 1 bit

The PanInfoExists bit specifies when the panning information is updated in each sub block boundary. The panning information always exists for the first sub block of a frame. The decoder should assume that if the PanInfoExists bit is set to zero then the panning information is repeated from the previous sub block.

5.4.3 ObjectPosX[sb], ObjectPosY[sb], ObjectPosZ[sb] – 16 bits

The ObjectPosX, ObjectPosY, and ObjectPosZ codes specifiy the 3 dimensional position of a panned audio object. The ObjectPos field is encoded using the position coding method described is Section 2.

5.4.4 ObjectSnap[sb] – 1 bit

The ObjectSnap[sb] code is a Boolean value, if it set to 1 then the object should snap to the closest speaker.

5.4.5 Zone IDs – Informative

The ZoneIDs are listed in Table 8.

Table 8 – Zone Ids

Zone ID	Description
0	All screen speakers left of center
1	Screen center speakers
2	All screen speakers right of center
3	All speakers on left wall
4	All speakers on right wall
5	All speakers on left half of rear wall
6	All speakers on right half of rear wall
7	All overhead speakers left of center
8	All overhead speakers right of center

5.4.6 ObjectZoneControl[sb] – 1 bit

The ObjectZoneControl[sb] is a Boolean value; if the value is set to 0, zone control is not used, else control information follows. If present, the control information shall be included in the bitstream for every zone listed in Table 8, in the order listed in Table 8.

5.4.7 ZoneGain – 2 bits

The meaning of the code is shown in Table 9. ZoneGain shall be associated with a Zone according to the order it appears in the bitstream.

Table 9 – Zone Gain Code

Code	Gain
0x0	Set gain to 0.0
0x1	Set gain of 1.0
0x2	Reserved
0x3	Reserved

5.4.8 ObjectSpreadMode – 2 bits

The ObjectSpreadMode code specifies what type spreading shall be applied to the object as shown in Table 10.

Table 10 – Object Spread Mode Code

Code	Identifier	Description
0x0	OBJECT_SPREAD_LOWREZ	Equal spreading in all dimension using 8-bit coding
0x1	RESERVED	Reserved
0x2	OBJECT_SPREAD_1D	Equal spreading in all dimensions using 12-bit coding
0x3	RESERVED	Reserved

5.4.9 ObjectSpread[sb] – 8 or 12 bits

The ObjectSpread[sb] code shall specify the amount of spread to be applied to an object and corresponds to the extent of spread in each direction (X, Y, or Z). The code is converted to a value ranging between 0.0 and 1.0 using the DistanceZ coding method described in Section 3.2. ObjectSpread is the spread of the object in each dimension; i.e., the spread corresponds to +/- 0.5 * ObjectSpread from the object's position.

5.4.10 ObjectDecorCoefPrefix – 2 bits

This element is used to simplify the coding of common gain values. The meaning of the code is shown in Table 11.

Table 11 – Decorrelation Coef Prefix Code

Code	Gain
0x0	No decorrelation
0x1	Maximum decorrelation
0x2	Decorrelation coefficient follows in the bitstream.
0x3	(Reserved)

5.4.11 ObjectDecorCoef – 8 bits

The ObjectDecorCoef code specifies the amount of decorrelation to be applied to the bed channel's audio asset. ObjectDecorCoef = 0 means no decorrelation. ObjectDecorCoef = 255 means maximum decorrelation.

5.5 AudioDataDLC() Data Fields

5.5.1 DLCSize – 16

DLCSize shall indicate the size in bytes of the remainder of the AudioDataDLC Element, or equivalently, the size of the entire element, not including ElementID, ElementSize, AudioDataID, and DLCSize.

5.5.2 DLCSampleRate – 2 bits

The DLCSampleRate shall specify the sample rate of the audio contained in the AudioDataDLC element. The meaning of the code is shown in Table 12.

Table 12 - Sample Rate code

DLCSampleRate Code	Meaning
0x0	48000 samples per second
0x1	96000 samples per second
0x2	Reserved
0x3	Reserved

5.5.3 ShiftBits – 5 bits

The ShiftBits code specifies the number of least significant bits that should be added to each output audio sample in the decoder.

5.5.4 NumPredRegions – 2 bits

The NumPredRegions code specifies the number of predictor regions within an audio frame. Each predictor region is a group of predictor region blocks. The number of predictor region blocks is dependent on the sample rate and frame rate are shown in Table 13.

Table 13 – Predictor Region Blocks versus Sample Rate and Frame Rate

Sample Rate	Frame Rate	Total Predictor Region Blocks Per Frame	Predictor Region Block Size (Samples)
48 kHz	24	10	200
48	25	10	192
48	30	8	200
48	48	5	200
48	50	5	192
48	60	4	200
48	96	5	100
48	100	4	120
48	120	4	100
96 kHz	24	10	400
96	25	10	384
96	30	8	400
96	48	5	400
96	50	5	384
96	60	4	400
96	96	5	200
96	100	4	240
96	120	4	200

5.5.5 RegionLength – 4 bits

The RegionLength code specifies the number of predictor region blocks in a predictor region.

5.5.6 FIROrder – 5 bits

The FIROrder code specifies the filter order for the FIR predictor. A value of 0 indicates the filter is not used.

5.5.7 FIRPredictor – 10 bits

The FIRPredictor is the quantized forward prediction coefficients. The prediction coefficients are transmitted as refection coefficients.

5.5.8 NumSubBlocks, SubBlockSize – Informative

The number of sub blocks (NumSubBlocks) and sub block size (SubBlockSIze) are determined by the sample rate and frame rate as shown in Table 14.

Table 14 – Number of Sub Blocks and Sub Block Size versus Sample Rate and Frame Rate

Sample Rate	Frame Rate	NumSubBlocks	SubBlockSize (samples)
48 kHz	24	10	200
48	25	10	192
48	30	8	200
48	48	5	200
48	50	5	192
48	60	4	200
48	96	5	100
48	100	4	120
48	120	4	100

96 kHz	24	10	400
96	25	10	384
96	30	8	400
96	48	5	400
96	50	5	384
96	60	4	400
96	96	5	200
96	100	4	240
96	120	4	200

5.5.9 CodeType – 1 bit

The CodeType code specifies the entropy coding technique applied to the residual signal as specified in Table 15.

Table 15 – Residual Coding Type

Code	Residual Coding Type
0x0	Direct PCM
0x1	Rice-Golomb Coding

5.5.10 BitDepth – 5 bits

If direct PCM is used to transmit the residual signal, then the BitDepth code specifies the number of bits used to represent the PCM data in the sub block.

5.5.11 RiceCode – 5 bits

If Rice-Golomb coding is used to transmit the residual signal then the RiceCode code controls the Rice-Golomb entropy decoding.

5.5.12 Residual[n * SubBlockSize + l] – Variable

The Residual[n * SubBlockSize + l] data is the residual signal for the frame of audio either in direct PCM or in Rice-Golomb code.