
**Information technology — High
efficiency coding and media delivery
in heterogeneous environments —**

**Part 3:
3D audio**

**AMENDMENT 3: MPEG-H 3D Audio
Phase 2**

*Technologies de l'information — Codage à haute efficacité et livraison
des médias dans des environnements hétérogènes —*

Partie 3: Audio 3D

AMENDEMENT 3: Phase 2 de l'audio 3D MPEG-H





COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2017, Published in Switzerland

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
Ch. de Blandonnet 8 • CP 401
CH-1214 Vernier, Geneva, Switzerland
Tel. +41 22 749 01 11
Fax +41 22 749 09 47
copyright@iso.org
www.iso.org

Contents

Page

Foreword	v
Introduction.....	vi
1 Profiles and Levels	7
2 Technical Overview - Update.....	18
3 MPEG Surround	21
4 3D Audio Phase II – HOA (Subband Directional Prediction, Parametric Ambiance Replication, Phase-based Decorrelation, HOA Layered Coding)	23
5 Optimizations and Improvements for Low Bitrate Coding.....	125
6 Joint Channels for Low Bitrate Coding	163
7 Discrete Multi-Channel Coding Tool	173
8 Updates to MHAS	190
9 Metadata Updates	197
9.1 Update of mae_Data() syntax and semantics	197
9.2 Update of OAM data transmission and processing	203
9.2.1 OAM syntax and semantics	203
9.2.2 2D spread rendering.....	218
9.2.3 Informative distance and depth spread rendering.....	220
9.3 Signaling and Processing of Scene Displacement Angles for CO content	221
9.4 Extension of screen-related processing for off-centered screens	230
9.5 Update of closest speaker ployout for the conditioned case	235
9.6 Processing of excluded sectors	237
9.7 Interface for channel-based, object-based, and HOA metadata and audio.....	238
9.8 Diffuseness Rendering	249
9.8.1 Diffuseness Processing.....	249
9.8.2 Informative decorrelation filtering for diffuseness processing.....	252
9.9 Updates of the element metadata preprocessor	253
9.10 Review of Metadata	262
9.11 References	271
10 Improvements for use in broadcast ecosystems.....	271
10.1 Order of elements in mpeg3daDecoderConfig() and mpeg3daFrame()	271
10.2 Overall delay alignment and constant decoder delay	273
10.3 Broadcast Contribution Mode Operation of MPEG-H	276
10.4 Audio Pre-Roll.....	277
10.5 Multi-stream Handling.....	284
11 SAOC signaling update.....	287
12 Tool for Advanced Loudness Control	289
13 Frequency-Domain Prediction and Time-Domain Post-Filtering.....	293
14 Sample Rate Converter	302
15 Low Complexity Downmix	303
16 Tonal Component Coding.....	378
17 Internal Channel on MPS212 for Low Complexity Format Conversion	390
18 Low Complexity HOA Spatial Decoding and Rendering	403
19 High Resolution Envelope Processing (HREP)	417

20	Signaling of IGF start and stop bands.....	428
21	Consolidated Tables for Configuration Extensions, mpeg3daConfigExtension(),usacConfigExtType	430
22	Consolidated Tables for Extensions Element Configuration and Payload, mpeg3daExtElementConfig(),usacExtElementType	432
23	Consolidated Tables for MAE Data Types, mae_data(), mae_dataType	435
24	Consolidated Table for tcx_coding()	437
25	Peak Limiter.....	439
26	Informative Annex on screen-related adaptation of HOA content in complexity constrained implementations.....	441
27	Further Changes, Not Categorized	442
28	Retaining original file length with MPEG-H 3D Audio.....	447
	AMD.OFL.1General.....	447
	AMD.OFL.2 Avoiding Leading Zero Sampl	447
	AMD.OFL.3 Avoiding Trailing Zero Samples.....	448
29	Enhanced Noise Filling	449
30	Scope	453
31	Main Profile.....	454

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation on the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see the following URL: www.iso.org/iso/foreword.html.

Amendment 3 to ISO/IEC 23008-3:2015 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

Introduction

The following text describes the Amendment 3 to the specification ISO/IEC 23008-3:2015 MPEG-H 3D Audio in an "Amendment"-style, i.e. in "Replace A with B"-style. It includes additions and changes that serve a number of purposes:

- improving the coding efficiency especially for low bitrate coding modes (for scene based as well as for object based and for multichannel based content)
- adding descriptive metadata
- updating the MHAS description
- some improvements for usage of MPEG-H 3D Audio in broadcasting applications
- a tool for Advanced Loudness Control
- a layered coding mode for coding of scene based content

It is envisioned that this amendment will be merged with the current version of the MPEG-H 3D Audio specification resulting in a second edition of the standard. Text with yellow highlight shall be adjusted by the editor of a new edition of ISO/IEC 23008-3.

For ease of review the document is structured by clauses, each of which reflect an approved set of changes.

New Clauses, Tables and Figures are typically labelled "AMD_X.Y", where X is the number of the clause it appears in in this document and Y is an increasing integer counter.

Information technology — High efficiency coding and media delivery in heterogeneous environments — Part 3: 3D audio, AMENDMENT 3: MPEG-H 3D Audio Phase 2

1 Profiles and Levels

Add the following definition of profiles and levels to clause 4 Technical Overview:

4.X MPEG-H 3D Audio profiles and levels

4.X.1 Introduction

This subclause defines profiles and their levels for MPEG-H 3D Audio.

Complexity units are defined to give an approximation of the decoder complexity in terms of processing power required for the decoding process. The approximated processing power is given in "Processor Complexity Units" (PCU), specified in Millions Operations Per Second (MOPS).

4.X.2 Profiles

The following Audio Profiles are defined:

1. The Main Profile of MPEG-H 3D Audio provides a complete set of features for low-bitrate and high-quality coding, and rendering for all playback scenarios, exclusively based on the first edition of the MPEG-H 3D Audio specification ISO/IEC 23008-3:2015 3D Audio.
2. The High Profile of MPEG-H 3D Audio provides a complete set of features for low-bitrate and high-quality coding, and rendering for all playback scenarios.
The High Profile is a superset of the Low-complexity Profile.
3. The Low Complexity Profile provides features for broadcasting and streaming with a reduced complexity of the decoder;

Table P1 — Summary of the Location of and Normative Reference to the Definitions of MPEG-H 3D Audio profiles. USAC and MPEG-H 3DA Main Profile are provided for information only

Tool / Module		defined in ISO/IEC	sub-clause	USAC 23003-3	MPEG-H 3DA Main Profile	MPEG-H 3DA High Profile	MPEG-H 3DA Low-Complexity Profile
block switching		14496-3	4.6.11	X	X	X	X
window shapes	AAC based	14496-3	4.6.11	X	X	X	X
	additional windows	23003-3	6.2.9.3	X	X	X	X
filter bank	AAC based	14496-3	4.6.11	X	X	X	X
	additional USAC	23003-3	7.9	X	X	X	X

TNS		14496-3	4.6.9	X	X	X	X
intensity		14496-3	4.6.8.2				
coupling		14496-3	4.6.8.3				
perceptual noise synthesis	PNS	14496-3	4.6.13				
	noise filling	23003-3	7.2	X	X	X	X
MS	basic mid/side coding	14496-3	4.6.8.1	X	X	X	X
	MDCT based complex prediction	23003-3	7.7.2	X	X	X	X
quantization	non-uniform	14496-3	4.6.1	X	X	X	X
	uniform	23003-3	7.1	X	X	X	X
entropy coding	Huffman	14496-3	4.6.3				
	context adaptive arithmetic coding	23003-3	7.4	X	X	X	X
SBR	base	14496-3	4.6.18	X	X	X	
	enhanced	23003-3	7.5	X	X	X	
parametric stereo extension	Parametric Stereo	14496-3	8.6.4 / 8.A				
	MPEG Surround 2-1-2 (incl. residual coding)	23003-3	6.2.13	X	X	X	
	Quad Channel Element	23008-3	5.5		X	X	
ACELP		23003-3	7.14	X	X	X	X
frequency domain noise shaping	scale factor based	14496-3	4.6.2	X	X	X	X
	LPC based	23003-3		X	X	X	X
Intelligent Gap Filling	IGF for FD	23008-3			X	X	X
Improved LPD coding	IGF for TCX and TBE in ACELP	23008-3 Amd3				X	X
	LPD stereo	23008-3 Amd3				X	X
Predictors for FD and TCX	frequency-domain prediction and time-domain post-filtering	23008-3 Amd3				X	X
Discrete Multi-channel coding	MCT	23008-3 Amd3				X	X
Format Converter	Generic downmix	23008-3	10, Amd3.1		X	X	X (Note 4)
Immersive Rendering	Immersive rendering within format converter	23008-3	11, Amd3.2		X	X	X (Note 4)
Static metadata	Metadata Audio Elements (MAE) and Audio Scene Information (ASI) Decoder and Renderer	23008-3	15		X	X	X
Dynamic object metadata	Object Audio Metadata (OAM) Decoder and Renderer	23008-3	7, 8		X	X	X
	MPEG Surround Extension	23003-1 Amd 3	9			X	
SAOC-3D	Decoder and Renderer	23008-3	9		X	X	
HOA	Decoder and Renderer	23008-3 and Amd3	12		X	X	X (Note 5)

	Near Field Compensation	23008-3			X	X	X (Note1)
	Subband Directional Prediction	23008-3 Amd3				X	
	Parametric Ambiance Replication (PAR)	23008-3 Amd3				X	
	Phase-based decorrelation	23008-3 Amd3				X	
Binaural	FD-binaural, TD-binaural	23008-3	13		X	X	X (Note2)
	HOA2Binaural H2B	23008-3			X	X	X (Note2)
DRC	DRC-1	23003-4			X	X	X (Note3)
	DRC-2 (single band)	23003-4			X	X	X
	DRC-2 (multi band)	23003-4					
	DRC-3 (single band)	23003-4			X	X	X
Sample Rate Converter		23008-3 Amd3	Amd3.3			X	X
Peak Limiter	Unguided clipping prevention	23008-3 23003-4	D			X	X
Loudness	Loudness metadata and handling	23003-4	6		X	X	X
	Loudness compensation	23008-3 Amd3				X	X
MHAS	MPEG-H 3D audio stream	23008-3	14		X	X	X
	Truncation message and CRC packet type, ASI packet type	23008-3 Amd3				X	X
File Format	Carriage of MPEG-H 3D Audio in ISO base media file format	23008-3 Amd2			(Note 6)		
Interfaces and processing	Interfaces and processing for Interaction data and local setup info	23008-3	17,18		X	X	X
Carriage of system data	Carriage of System Data for the interaction with System Engine	23008-3 Amd4				X	X
TCC	Tonal Component Coding	23008-3 Amd3				X	
IC	Internal Channel	23008-3 Amd3				X	
HREP	High Resolution Envelope Processing	23008-3 Amd3				X	

Note 1: Restrictions apply dependent on the levels

Note 2: Implementation of binaural rendering is only mandated if headphone reproduction is supported.

Note 3: Multi-band DRC-1 shall be applied in the STFT domain of the TD format converter.

Note 4: The TD format converter downmix shall be applied for downmixing.

Note 5: In order to achieve target complexity for the LC profile at a given level implementers should study Annex G.

Note 6: File Format encapsulation is independent of the profile that is used for the bitstream. A profile level indicator is part of the file format specification (see XXX).

4.X.2.1 Levels of the Low Complexity Profile

Table P2 — Levels and their corresponding restrictions for the Low Complexity Profile

Level	Max. Sampling rate	Max. no. of core ch. in compressed data stream	Max. no. of decoder processed core ch.	Max. no. of loudspeaker output ch.	Example of max. loudspeaker configuration	Max. no. of decoded objects	Example of a max. Config C+O	Max. HOA order	Example of max. HOA order + O
1	48000	10	5	2	2.0	5	2 ch. + 3 static obj. ^{NOTE 1}	2	2 nd order + 3 static obj. ^{NOTE 1}
2	48000	18	9	8	7.1	9	6 ch. + 3 static obj. ^{NOTE 1}	4	4 th order + 3 static obj. ^{NOTE 1}
3	48000	32	16	12	11.1	16	12 ch. + 4 obj.	6	6 th order + 4 obj.
4	48000	56	28	24	22.2	28	24 ch. + 4 obj.	6	6 th order + 4 obj.
5	96000	56	28	24	22.2	28	24 ch. + 4 obj.	6	6 th order + 4 obj.

NOTE 1 – In this context "static objects" are understood as channel-based signals without accompanying OAM data which are not also associated to a channel bed

- The use of switch groups determines the subset of core channels out of the core channels in the bitstream that shall be decoded.
- If the mae_AudioSceneInfo() contains switch groups (mae_numSwitchGroups>0), then the elementLengthPresent flag shall be 1
- The number of channels of the signaled referenceLayout shall not exceed the maximum number of loudspeaker output channels as defined in the levels **Table P2**

Table P3 — Approximated worst case processing power (PCU) of decoder modules and the whole decoder for the different Levels of the Low Complexity Profile given in MOPS

Level	Core LC	Format Converter	Object Renderer	HOA ²	Objects only Renderer	DRC	Limiter	Binaural ¹	Worst case PCU
1	33	3	0	3	9	6	4	7	58
2	59	10	0	17	16	18	5	19	118
3	106	36	7	36	29	24	6	27	206
4	186	113	7	93	50	30	9	46	392
5	373	226	14	186	50	34	19	92	758

¹ NOTE: The complexity numbers for binaural processing are calculated on the basis of BRIR filters of 1 second length measured in a BS.1116 compliant room.

² NOTE: The complexity numbers for the HOA spatial decoding and rendering are based on the Low

Complexity Combined HOA Spatial Decoding and Rendering described in Annex G.

4.X.2.2 Restrictions for the Low Complexity Profile and Levels

In the low complexity profile the core decoder, format converter, object renderer, HOA renderer and DRC and peak limiter operate in time domain, MDCT-domain or STFT-domain.

The following restrictions apply for HOA renderer and decoder:

Table P4 — Restrictions for the HOA spatial Decoding and Rendering according to the Level of the Low Complexity Profile

Restriction applies to	Maximum allowed value depending on Mpeg3daProfileLevelIndication				
	Lvl 1	Lvl 2	Lvl 3	Lvl 4	Lvl 5
HOA order (max)	2	4	6	6	6
Number of Predominant Sounds (max)	3	5	7	8	8
Number of directional signals used in prediction (max)	2	3	4	5	5
The Near Field Compensation (NFC) processing may be applied to HOA content of an order which is smaller or equal to:	N/A (NFC not allowed)	1	2	3	3

NFC may be employed in not more than one signal group of type SignalGroupTypeHOA.

The following restrictions apply to MPEG-D DRC (ISO/IEC 23003-4) when employed as part of MPEG-H 3D audio:

- drcFrameSizePresent and timeDeltaMinPresent shall be set to 0.
- gainInterpolationType shall be set to 1.
- dependsOnDrcSetPresent shall be set to 0 for drcInstructionsUniDrc() with downmixId == 0.
- HOA signal groups shall be restricted to one drcChannelGroup and DRC gains shall be applied to the HOA core channels (HOATransportChannels).
- The values of bsSequenceIndex within drcInstructionsUniDrc() shall be unique in simultaneously applied DRC sets except for bsSequenceIndex == 0.
- Multiband DRC shall be restricted to drcInstructionsUniDrc() with downmixId == 0. If the bitstream should contain multiband DRC, the number of multiband DRC core channels shall be restricted as follows:

$$\begin{aligned}
 &(\text{numAudioChannels} + \\
 &\text{numAudioObjects} \quad + \text{numAudioObjectsMB} + \\
 &\text{numHOATransportChannels} + \text{numHOATransportChannelsMB}) \\
 &\leq (\text{numCoreChannelsMax(Lvl)} - \text{dependsOnDrcSetPresentFlag} - 1)
 \end{aligned}$$

, where:

- numAudioChannels, numAudioObjects and numHOATransportChannels are the number of C, O and HOA core channels as specified in Table 4.

- numAudioObjectsMB and numHOATransportChannelsMB are the number of O and HOA core channels out of numAudioObjects and numHOATransportChannels that contain multiband DRC
- numCoreChannelsMax is the maximum number of core channels ("No of Core ch") depending on the MpegH3daProfileLevelIndication field as defined in [Table P2](#)
- dependsOnDrcSetPresentFlag is set to one if the bitstream contains any configuration with dependsOnDrcSetPresent==1 (otherwise zero).
- nNodes shall be restricted to a maximum value of 32, where nNodes is the number of encoded gain values in the current DRC frame.

Table P5 — Restrictions applying to DRC processing according to the Levels of the Low Complexity Profile

Restriction applies to	Maximum allowed value depending on MpegH3daProfileLevelIndication				
	Lvl 1	Lvl 2	Lvl 3	Lvl 4	Lvl 5
nDrcChannelGroupsTotal (Note 1)	5	9	16	28	28
drcCoefficientsUniDrcCount	4	4	4	4	4
bandCount (Note 2)	2	4	4	4	4
sequenceCountTotal (Note 3)	24	28	32	48	63
drcInstructionsUniDrcCount	16	16	32	32	32
Note 1: Maximum allowed number of simultaneously active DRC channel groups in all applied DRC sets.					
Note 2: Maximum allowed number of DRC bands for multiband DRC.					
Note 3: Sum of all nDrcBands in drcGainSequence() structures plus number of sequences with gainCodingProfile=3.					

The following tool specific restrictions apply:

- If the independent noise filling (INF) of the intelligent gap filling (IGF) is activated (i.e. if igfUseEnf==1), then the Complex Prediction tool shall be restricted to real-only prediction, i.e. complex_coef shall be 0.
- If Stereo Filling is activated (i.e. if stereo_filling==1), then the Complex Prediction tool shall be restricted to real-only prediction, i.e. complex_coef shall be 0.
- The independent noise filling of the intelligent gap filling shall not be employed in cases where igfBgn corresponds to an audio frequency higher than 8 kHz.
- The LPD mode shall only be employed at 3DA core coder sampling rates (as defined in [Table 2 — Syntax of mpegH3daConfig\(\)](#)) ≤ 32000 Hz

EXAMPLE For a 48 kHz input signal, the encoder resamples the signal to a 32 kHz core coder sampling rate and the LPD decoder operates at this lower sampling rate. After the core decoding the signal is resampled to 48 kHz.

- The multi-channel coding tool (MCT) shall not employ more stereo boxes than specified in [Table P9](#)

Table P9 — Restrictions applying to MCT processing according to the Levels of the Low Complexity Profile

Restriction applies to	Maximum allowed value depending on MpegH3daProfileLevelIndication				
	Lvl 1	Lvl 2	Lvl 3	Lvl 4	Lvl 5
Number of stereo boxes in MCT	5	9	16	28	28

The following restrictions apply to coding of audio objects and the associated OAM data:

Table P10 — Restrictions applying to Object Processing According to the Levels of the Low Complexity Profile

Restriction applies to	Maximum allowed value n depending on MpegH3daProfileLevelIndication				
	Lvl 1	Lvl 2	Lvl 3	Lvl 4	Lvl 5
(number of objects without divergence) + $3 \cdot (\text{number of objects with divergence} > 0) \leq n$	5	9	16	28	28

- Efficient Object Metadata Decoding is not permitted, i.e. lowDelayMetadataCoding shall be 1.
- Furthermore the OAM frame length shall comply to:

$$\text{OAMFrameLength} = \text{outputFrameLength} / n,$$
with n being a positive integer in the range of $\{1, \dots, 4\}$
- Objects shall not employ divergence and spread at the same time.
 - If an object is defined with a spatial extent (spread $\alpha > 0.0^\circ$ for uniform spread, spread_width $\alpha_{\text{width}} > 0.0^\circ$ for non-uniform spread) it shall have a divergence value equal to zero.
 - If an object is defined with a divergence value > 0 , it shall not have a spatial extent (spread α shall be equal to 0.0° for uniform spread, spread_width α_{width} shall be equal to 0.0° for non-uniform spread)

The following restrictions apply to binaural rendering:

The value of bsBinauralDataFormatID in BinauralRendering() should be set to 1 (if the FD Binaural renderer is implemented) or to 2 (if the TD Binaural renderer is implemented). The value of bsBinauralDataFormatID can be set to 0 if the Parameterization of Binaural Room Impulse Responses according to 13.2.3 or 13.3.3 is implemented.

The number of BRIR sets is restricted to a maximum number of 3.

In case of H2B filters, the number of BRIR filter pairs to be provided shall correspond to 'Maximum H2B filter order' column in Table P8. In the other cases, the following applies:

The number of BRIR pairs in each BRIR set shall correspond to the number indicated in the relevant level-dependent row of Table P8. The measured BRIR positions shall correspond to all nominal geometric positions corresponding to the list of LoudspeakerGeometry indices in Table P8. The correspondence between LoudspeakerGeometry index and nominal geometric position is defined in ISO/IEC 23001-8. Thereby, it is ensured that one BRIR pair is available for each possible regular input channel configuration that can be used within the indicated level.

An input channel configuration is regular if it is defined by means of an ISO/IEC 23001-8 ChannelConfiguration or a list of ISO/IEC 23001-8 LoudspeakerGeometry (CICPSpeakerIdx).

If binaural rendering is activated, the measured BRIR positions shall be passed to the mpegH3daLocalSetupInformation(). Thus, all renderer stages are set to the target layout that is equal to the transmitted channel configuration. As one BRIR is available per regular input channel, the Format Converter can be passed through in case regular input channel positions are used.

Table P8 — The binaural restrictions for the LC profile

Level	Number of BRIR pairs	Maximum H2B filter order	BRIR positions by means of Loudspeaker Position Abbreviation	BRIR positions by means of Loudspeaker Geometry according to ISO/IEC 23001-8
1	3	1	L, R, C	0, 1, 2
2	10	2	L, R, C, LS, Rs, Lc, Rc, Lsr, Rsr, Cs	0, 1, 2, 4, 5, 6, 7, 8, 9, 10,
3	21	3	L, R, C, Ls, Rs, Lc, Rc, Lsr, Rsr, Cs, Lss, Rss, Lv, Rc, Cv, Lvr, Rvr, Cvr, Rs, Lvs, Rvs	0, 1, 2, 4, 5, 6, 7, 8, 9, 10, 13, 14, 17, 18, 19, 20, 21, 22, 25, 30, 31
4	28	5	L, R, C, Ls, Rs, Lc, Rc, Lsr, Rsr, Cs, Lss, Rss, Lv, Rc, Cv, Lvr, Rvr, Cvr, Lvss, Rvss, Ts, Lb, Rb, Cb, Lvs, Rvs, Lbs, Rbs,	0, 1, 2, 4, 5, 6, 7, 8, 9, 10, 13, 14, 17, 18, 19, 20, 21, 22, 23, 24, 25, 27, 28, 29, 30, 31, 37, 38,
5	28	6	L, R, C, Ls, Rs, Lc, Rc, Lsr, Rsr, Cs, Lss, Rss, Lv, Rc, Cv, Lvr, Rvr, Cvr, Lvss, Rvss, Ts, Lb, Rb, Cb, Lvs, Rvs, Lbs, Rbs,	0, 1, 2, 4, 5, 6, 7, 8, 9, 10, 13, 14, 17, 18, 19, 20, 21, 22, 23, 24, 25, 27, 28, 29, 30, 31, 37, 38,

The following additional parameter values restrictions apply:

- The value of kMax in FdBinauralRendererParam() shall be equal to or less than 48 (bands).
- The value of kConv in FdBinauralRendererParam() shall be equal to 32.
- The values of rt60[k] in SfrBrirParam() shall be less than or equal to 1.0 (sec).
- The average of the values of nFilter[k] shall be less than or equal to 64.
- The values of nFilter[k] in VoffBrirParam() should be less than or equal to 256.

The following coding tools, modules, or features shall not be employed

- Time warped filterbank
- 768 sample outputFrameLength, i.e. coreSbrFrameLengthIndex shall not be 0

The following text describes restrictions dependent on the length of the arithmetic coder codeword, arith_data(). For this text the following definitions apply:

F_{sOut} core coder sampling rate as indicated by means of usacSamplingFrequencyIndex or usacSamplingFrequency in mpegH3daConfig()

F_{sMax} maximum allowed sampling rate of a given level in this profile

N_{chMax} maximum number of decoder processed core channels of a given level in this profile according to **Table P2**.

N_{chLtpf} number of core coder channels in which the long term post filter (LTPF) is applied

N_{chInf} number of core coder channels in which the independent noise filling (INF) is applied

$Nbits_{arith_data}(ch)$ number of bits used for arithmetic coding of spectral data, $arith_data()$, for core coder channel ch for a given frame

$Nbits_{arith_all} = \sum_{\text{all channels}} Nbits_{arith_data}(ch)$, i.e. the sum of all bits used for the arithmetic coding of spectral data of all core coder channel

— In any given audio frame $Nbits_{arith_all}$ shall comply with the following restriction:

$$Nbits_{arith_all} < \frac{(3072 \cdot N_{chMax} - 2048 \cdot N_{chLtpf} - 2048 \cdot N_{chInf}) \cdot F_{sMax}}{F_{sOut}}$$

The following restrictions apply to the AudioPreRoll() extension:

- Decoders conforming to this profile shall support the full decoding and correct handling of the AudioPreRoll() extension .
- The number of pre-roll frames, $numPreRollFrames$, in an AudioPreRoll() extension payload shall not exceed 1 (one).
- In access units that are embedded as pre-roll in an AudioPreRoll() extension the $usacExtElementPresent$ field for extensions of type $ID_EXT_ELE_AUDIOPREROLL$ shall be 0.

The following restrictions apply to the employed sampling rate and the resampler block:

- The sampling rate that is signaled by means of $usacSamplingFrequencyIndex$ or $usacSamplingFrequency$ shall be one of the values in the first column of [Table P6](#).
- Depending on the above mentioned sampling rate and the profile level the resampler may employ one of the resampling ratios indicated in [Table P6](#).

Table P6 — Allowed Sampling Rates and Resampling Ratios

Allowed sampling rate	Allowed resampling ratio depending on MpegH3daProfileLevelIndication				
	Lvl 1	Lvl 2	Lvl 3	Lvl 4	Lvl 5
96000	N/A	N/A	N/A	N/A	1
88200	N/A	N/A	N/A	N/A	1
64000	N/A	N/A	N/A	N/A	1.5
58800	N/A	N/A	N/A	N/A	1.5
48000	1	1	1	1	1 or 2
44100	1	1	1	1	1 or 2
32000	1.5	1.5	1.5	1.5	1.5 or 3
29400	1.5	1.5	1.5	1.5	1.5 or 3
24000	2	2	2	2	2
22050	2	2	2	2	2
16000	3	3	3	3	3
14700	3	3	3	3	3

The following restrictions apply to the coding of the audio scene information structure:

Table P7 — ProfileLevel dependent restrictions to selected fields of the mae_AudioSceneInfo()

Restriction applies to	Allowed maximum value depending on MpegH3daProfileLevelIndication				
	Lvl 1	Lvl 2	Lvl 3	Lvl 4	Lvl 5
mae_numGroups	5	9	16	28	28
mae_numSwitchGroups	2	4	8	14	14
mae_numGroupPresets	4	4	8	16	31
mae_bsGroupPresetNumConditions	5	9	16	16	16
mae_numDownmixIdGroupPresetExtensions per mae_groupPresetID	4	4	8	16	31
mae_bsNumDescLanguages	4	4	4	8	16
mae_bsDescriptionDataLength	256	256	256	256	256

— If mae_numSwitchGroups > 0, then elementLengthPresent shall be set to 1.

4.X.2.3 Levels of the Main Profile

Currently blank — Placeholder for Main Profile

4.X.2.4 Levels of the High Profile

Currently blank — Placeholder for High Profile

In 5.3.2 replace Table 39 — Value of mpegH3daProfileLevelIndication

Table 39 — Value of mpegH3daProfileLevelIndication

value	Indication of Profile	Indication of Level
0x00	reserved for ISO use	-
0x01-0xff	reserved for future profile definition	

with:

Table 39 — Value of mpegH3daProfileLevelIndication

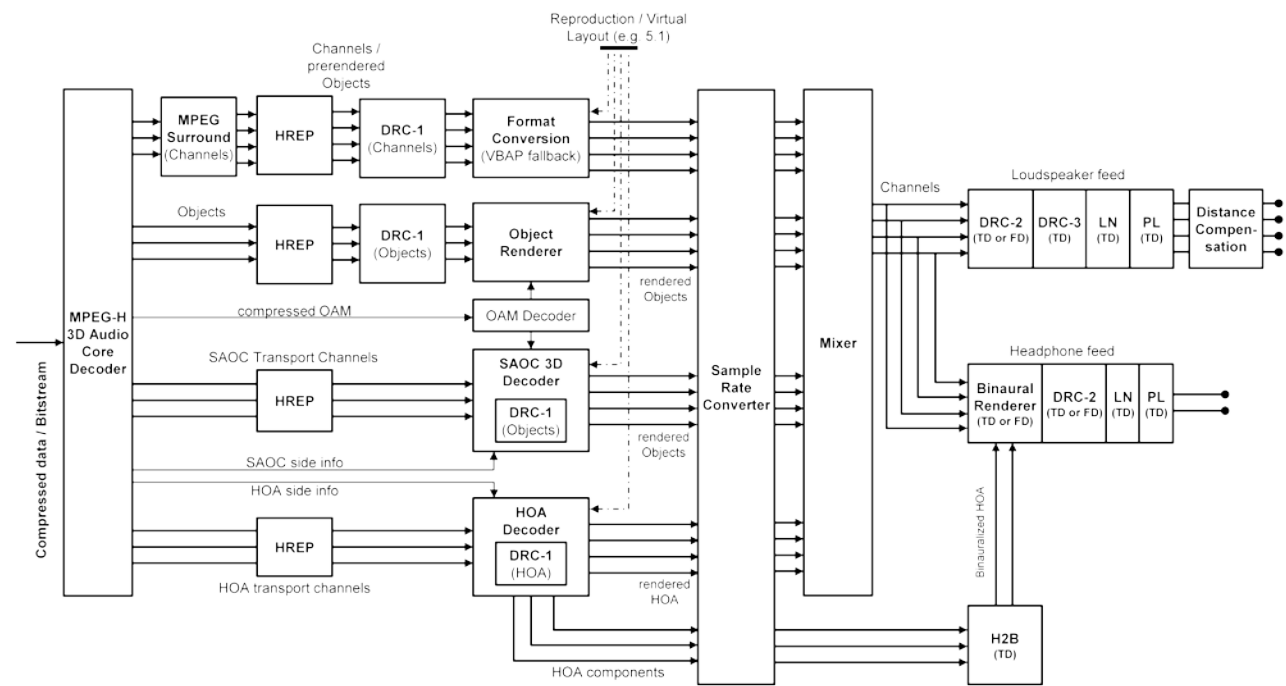
value	Indication of Profile	Indication of Level
0x00	reserved for ISO use	-
0x01	Main Profile	L1
0x02	Main Profile	L2
0x03	Main Profile	L3
0x04	Main Profile	L4
0x05	Main Profile	L5
0x06	High Profile	L1

0x07	High Profile	L2
0x08	High Profile	L3
0x09	High Profile	L4
0x0A	High Profile	L5
0x0B	Low Complexity Profile	L1
0x0C	Low Complexity Profile	L2
0x0D	Low Complexity Profile	L3
0x0E	Low Complexity Profile	L4
0x0F	Low Complexity Profile	L5
0x10-0xFF	reserved for future profile definition	

Please also study Clause 31 of this Amendment document

2 Technical Overview - Update

Replace Figure 1 — Block diagram of the 3D-Audio decoder. (DRC: Dynamic Range Control, SAOC: Spatial Audio Object Coding, HOA: Higher Order Ambisonics, LN: Loudness Normalization, PL: Peak Limiter) with:



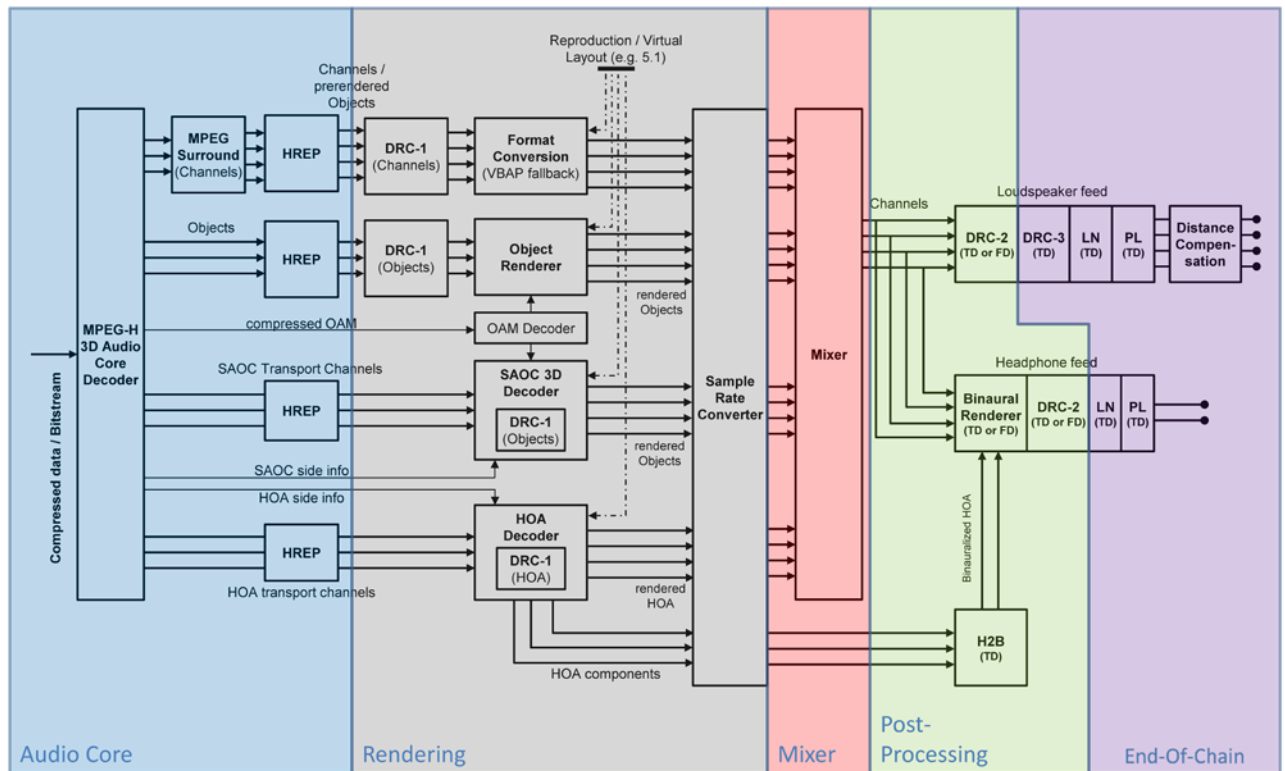
Add the following descriptive text after the description of the "HOA Decoder and Renderer" in clause 4.2:

The MPEG Surround Decoder takes the downmix signals coming from the MPEG-H 3D Audio decoder and performs the guided MPEG Surround upmix using the MPEG Surround side information to reproduce the multichannel signal for the transmitted loudspeaker layout.

In Table 1 — MPEG-H 3DA functional blocks and internal processing domain replace:

Audio Core	MPEG-H 3D Audio Core Coder	FD or TD ¹⁾
with:		
Audio Core	MPEG-H 3D Audio Core Coder	FD or TD ¹⁾
	MPEG Surround	FD

Replace Figure 2 — MPEG-H 3D audio decoder overview with signal processing context with:



In clause “4.2 Overview over the codec building blocks” after the first paragraph add following text and figure:

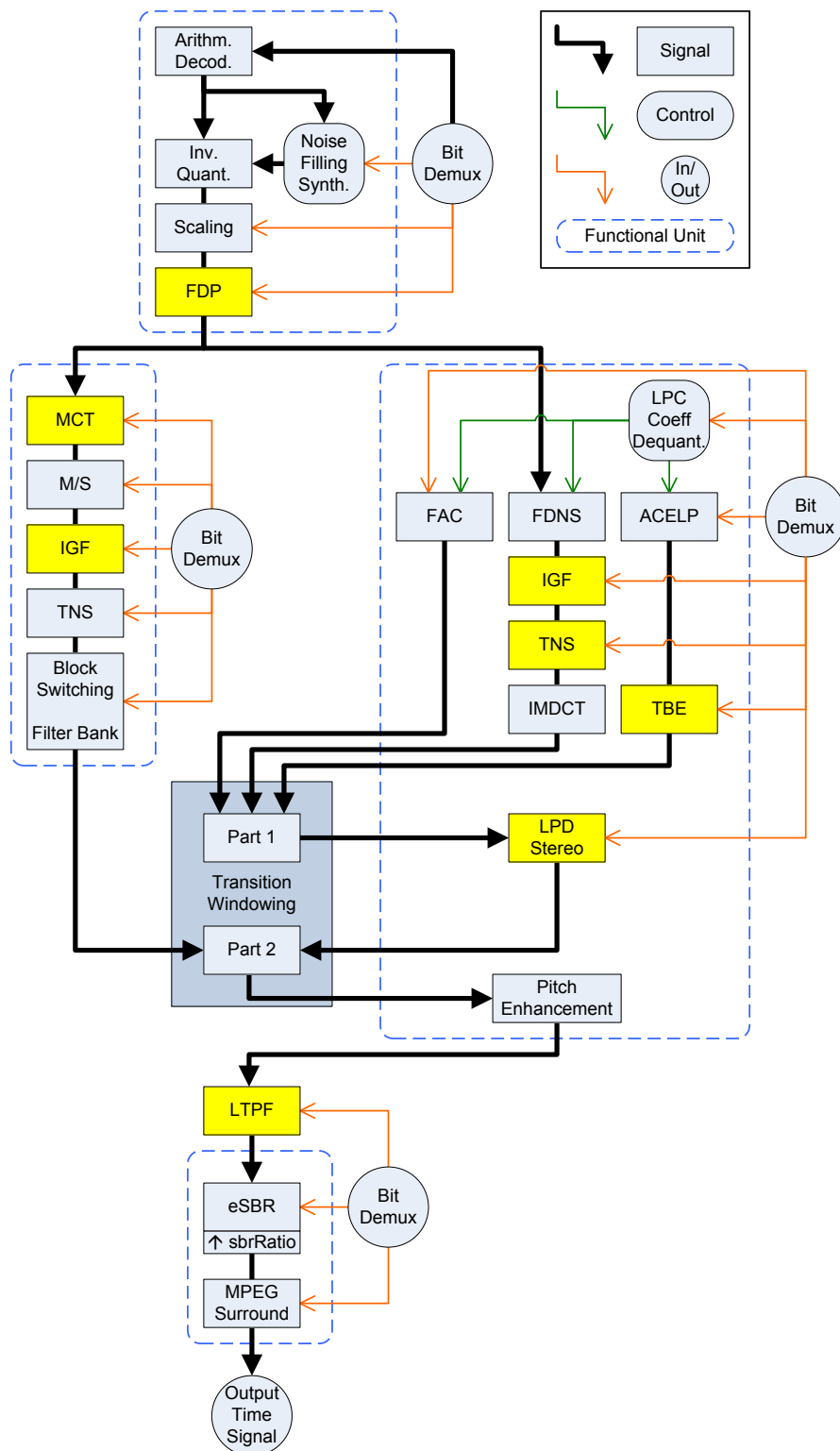


Figure AMD3.XX — Simplified block diagram of the typical MPEG-H core decoder configuration

Figure **AMD3.XX** shows a simplified block diagram of the typical MPEG-H core decoder building blocks. The major differences compared to MPEG-D USAC technology are highlighted in yellow. The highlighted tools are described in detail in section “MPEG-H 3D Audio Core decoder”.

3 MPEG Surround

Add the following new clause:

x MPEG Surround

x.1 Technical Overview

The output of the 3D Audio core decoder of the "channels / prerendered objects" path may be further processed by MPEG Surround (MPS). **Figure AMD1.1** shows a schematic of a combined 3D Audio core decoder and a MPS decoder.

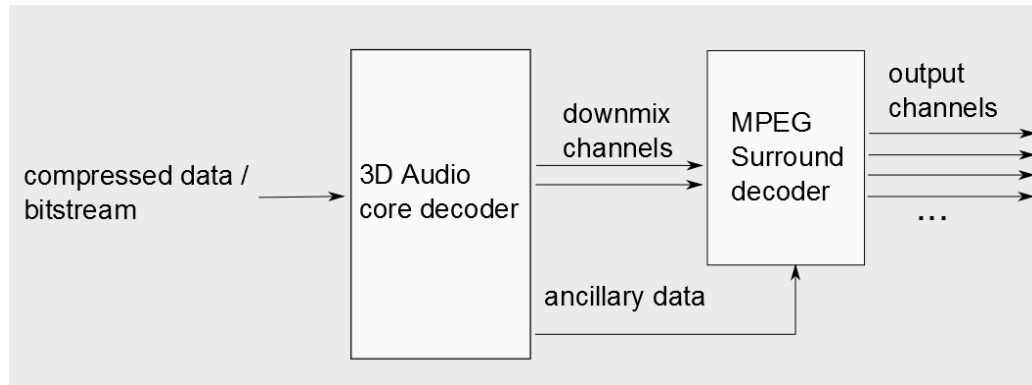


Figure AMD1.1: Block Diagram of a 3D Audio Core Decoder with an MPEG Surround decoder

If the SBR tool is active, a 3D Audio core decoder can typically be efficiently combined with a subsequent MPS decoder by connecting them in the QMF domain in the same way as it is described for USAC in clause 4.3 of ISO/IEC 23003-1:2012, *MPEG-D (MPEG audio technologies), Part 3: Unified Speech and Audio Coding, 2012*. If a connection in the QMF domain is not possible, the tools need to be connected in the time domain.

If MPS side information is embedded into a 3D Audio bitstream by means of the `usacExtElement` mechanism (with `usacExtElementType` being `ID_EXT_ELE_MPEGS`), the time-alignment between the 3D Audio data and the MPS data assumes the most efficient connection between the 3D Audio core decoder and the MPS decoder. If the SBR tool in the 3D Audio core decoder is active and if MPS employs a 64-band QMF domain representation (see 6.6.3 in ISO/IEC 23003-1:2007, *Information technology — MPEG audio technologies — Part 1: MPEG Surround*), the most efficient connection is in the QMF domain. Otherwise, the most efficient connection is in the time domain. This corresponds to the time-alignment for the combination of HE-AAC and MPS as defined in 4.4, 4.5, and 7.2.1 of ISO/IEC 23003-1:2007, *Information technology — MPEG audio technologies — Part 1: MPEG Surround*.

The additional delay introduced by adding MPS decoding after 3D Audio decoding is given by clause 4.5 of ISO/IEC 23003-1:2007, *Information technology — MPEG audio technologies — Part 1: MPEG Surround* and depends on whether HQ MPS or LP MPS is used, and whether MPS is connected to the 3D Audio core decoder in the QMF domain or in the time domain.

If multiple signal groups of type `SignalGroupTypeChannels` are present in the bitstream, one extension element conveying MPEG Surround data shall only refer to exactly only signal group. As per **5.3.1**, the corresponding channel elements shall directly follow that extension element.

If the MPEG Surround tool shall be used for one signal group of type `SignalGroupTypeChannels`, all core coder channels belonging to that signal group shall be fed to the MPEG Surround tool.

x.2 Syntax and Data Structure

The bitstream syntax and data structure are identical to the definitions in ISO/IEC 23003-1:2007, *Information technology — MPEG audio technologies — Part 1: MPEG Surround* and ISO/IEC 23003-1:2007/Amd.3:201x, *MPEG Surround extension for 3D Audio*.

x.3 Tool Description

The processing of the MPEG Surround tools are fully specified in ISO/IEC 23003-1:2007, *Information technology — MPEG audio technologies — Part 1: MPEG Surround* and ISO/IEC 23003-1:2007/Amd.3:201x, *MPEG Surround extension for 3D Audio*

4 3D Audio Phase II – HOA (Subband Directional Prediction, Parametric Ambiance Replication, Phase-based Decorrelation, HOA Layered Coding)

The following changes are related to HOA Matrix Encoder / Decoder

In Table 23 — Syntax of *HOARenderingMatrix()* replace:

precisionLevel	2	uimbsbf
if (gainLimitPerHoaOrder) {	1	uimbsbf

with:

maxHoaOrder = sqrt(NumOfHoaCoeffs)-1;		
precisionLevel	2	uimbsbf
isNormalized	1	uimbsbf
if (gainLimitPerHoaOrder) {	1	uimbsbf

In the same Table, replace

if (isFullMatrix) {	1	uimbsbf
firstSparseOrder	1	uimbsbf

with:

if (isFullMatrix ==0) {	1	uimbsbf
nbitsHoaOrder = ceil(log2(maxHoaOrder+1));		
firstSparseOrder	nbitsHoaOrder	uimbsbf

In the same Table, replace

} else { /* isAnyValueSymmetric==0 */		
if { isAnySignSymmetric } {	1	uimbsbf
for (i=0; i<numPairs; ++i)		
signSymmetricPairs[i] = boolVal	1	uimbsbf

with:

} else {		
if (isAnySignSymmetric) {	1	uimbsbf
for (i=0; i<numPairs; ++i)		
signSymmetricPairs[i] = boolVal ;	1	uimbsbf

In Table 25 — Syntax of *DecodeHoaMatrixData()* replace:

```

for (i = 0; i < inputCount; ++i) {
    for (j = 0; j < outputCount; ++j) {
        hoaMatrix[i * outputCount + j] *= signMatrix[i * outputCount + j];
        hoaMatrix[i * outputCount + j] /= sqrt(2 * ceil(sqrt(i+1)-1) + 1);
    }
}

```

with:

```

for (i = 0; i < inputCount; ++i) {
    for (j = 0; j < outputCount; ++j) {
        hoaMatrix[i * outputCount + j] *= signMatrix[i * outputCount + j];
        hoaMatrix[i * outputCount + j] /= sqrt(2 * ceil(sqrt(i+1)-1) + 1);
    }
}
if(isNormalized) {
    currentScalar = 0.0;
    for (i = 0; i < inputCount; ++i) {
        for (j = 0; j < outputCount; ++j) {
            if (!outputConfig[j].isLFE)
                currentScalar += hoaMatrix[i * outputCount + j] *
                                hoaMatrix[i * outputCount + j];
        }
    }
    currentScalar = 1.0/sqrt(currentScalar);
    for (i = 0; i < inputCount; ++i) {
        for (j = 0; j < outputCount; ++j) {
            if (!outputConfig[j].isLFE)
                hoaMatrix[i * outputCount + j] *= currentScalar;
        }
    }
}

```

In subclause 5.3.6 HOA Rendering Matrix Data Elements add before *precisionLevel*:

maxHoaOrder HOA Order of the transmitted matrix.

isNormalized Indicates if the HOA rendering matrix $\mathbf{D}$ is energy normalized, so that $\|\mathbf{D}\|_f = \sum_{l=1}^L \sum_{n=1}^{(N+1)^2} D_{l,n}^2 = 1$ with l being the non-LFE loudspeakers in the outputConfig.

In the same subclause before *firstSparseOrder*:

nbitsHoaOrder

Number of bits reading firstSparseOrder.

In Table 24 5.4.3.3 Decoding of HOA Rendering Matrix Coefficients after:

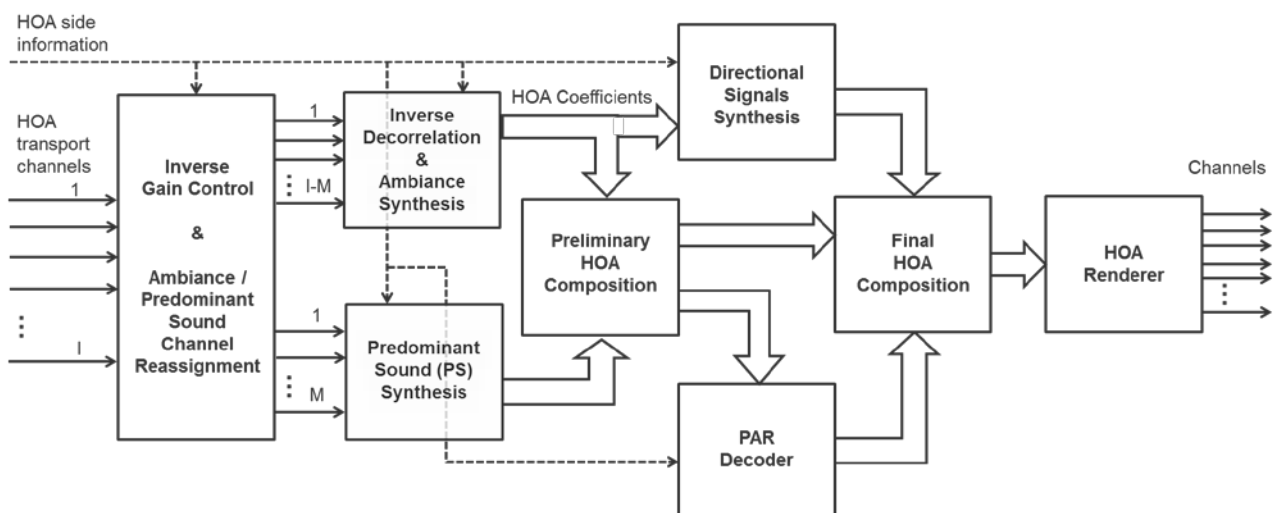
In this case the code words to decode the individual matrix elements for the left loudspeaker are reduced or completely omitted accordingly.

Add:

If the bitfield *isNormalized* was set to 1 the final HOA rendering matrix $\mathbf{D}$ is created by dividing each weighting value in the L rows of the HOA rendering matrix that are associated with non-LFE loudspeakers by the matrix's Frobenius Norm $\sum_{l=1}^L \sum_{n=1}^{(N+1)^2} D_{l,n}^2$ computed from its L rows associated with non-LFE loudspeakers.

The following changes are related to the chapter 12 Higher Order Ambisonics (HOA).

Replace Figure 33 — Simplified block diagram of the decoder with:



In subclause 12.1.1 Block Diagram replace:

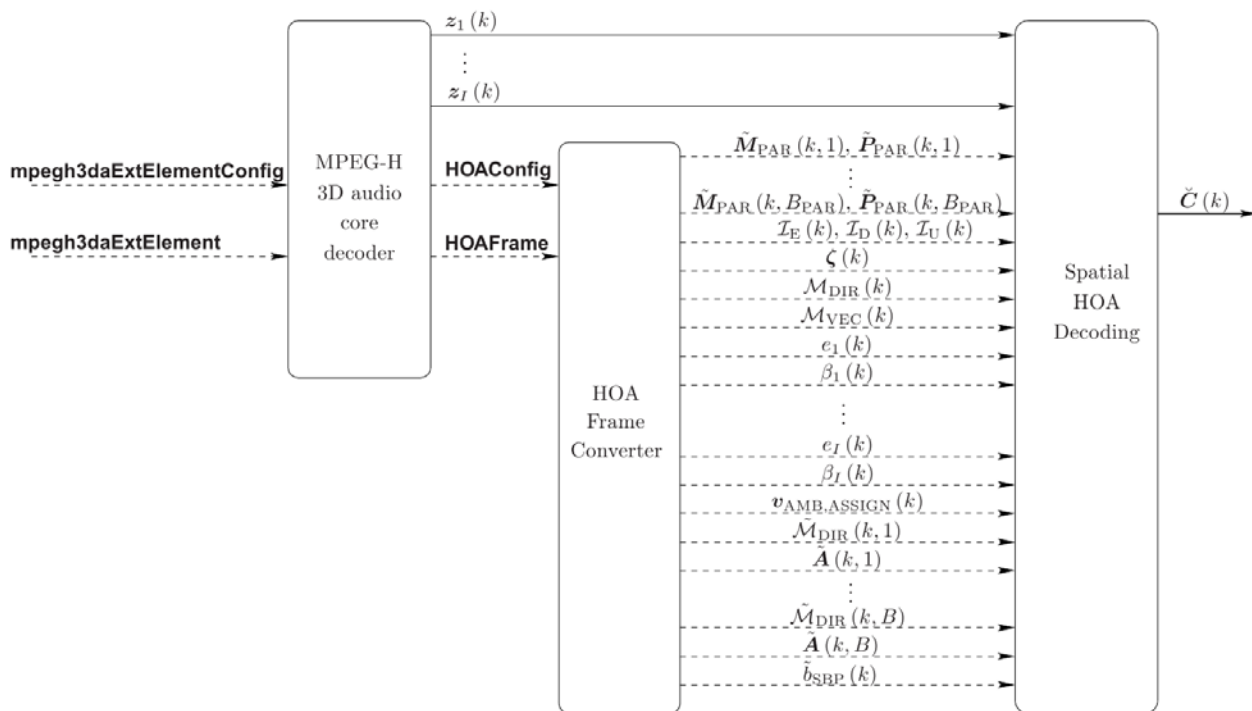
The fixed subset of the $(I - M)$ HOA coefficient signals is re-correlated, this means the decorrelation at the HOA encoding stage is reversed. Next, all of the $(I - M)$ HOA coefficient signals are used to create the ambient HOA components. The predominant HOA components are synthesized from the M predominant

signals and the corresponding parameters. Finally, the predominant and the ambient HOA components are composed into the desired full HOA representation, which is then rendered to a given loudspeaker setup.

with:

The fixed subset of the $(I - M)$ HOA coefficient signals is re-correlated, this means the decorrelation at the HOA encoding stage is reversed. Next, all of the $(I - M)$ HOA coefficient signals are used to create the ambient HOA component. The ambient HOA component is input to the Directional Signals Synthesis and the Preliminary HOA Composition. The Directional Signals Synthesis creates a new HOA representation from the ambient HOA component by prediction. The predominant HOA component is synthesized from the M predominant sound signals and the corresponding parameters. The predominant and the ambient HOA component are composed into the preliminary HOA representation, which is then input to the Parametric Ambience Replication (PAR). The PAR adds missing ambience components to the preliminary HOA representation, which is parametrically created from its input signals. Finally the output of the PAR, the preliminary HOA representation and the output of the Directional Signals Synthesis are composed to the decoded HOA representation that is rendered to the loudspeaker setup by the HOA Renderer.

Replace Figure 34 — The architecture of the HOA decoder tools with:



In subclause 12.1.2.1 HOA Decoding Tools replace:

- Ambient Synthesis
- HOA Compositions

with:

- Ambience Synthesis
- Preliminary HOA Composition
- Sub-band Directional Signals Synthesis

- Parametric Ambience Replication Decoder
- HOA Composition

Add new subclause 12.1.2.X "Layered Coding for HOA" after subclause 12.1.2.2 "HOA Renderer":

12.1.2.X Layered Coding for HOA

For the streaming of the compressed HOA sound field representation over a transmission channel with time-varying conditions layered coding is a means to adapt the quality of the received sound representation to the transmission conditions, and in particular to avoid undesired signal dropouts.

For layered coding the compressed HOA sound field representation is subdivided into a high priority base layer of a relatively small size and additional enhancement layers with decremental priorities and arbitrary sizes. Each enhancement layer is assumed to contain incremental information to complement that of all lower priority layers in order to improve the quality of the compressed HOA sound field representation. The idea is to control the amount of error protection for the transmission of the individual layers according to their priority. In particular, the base layer is provided with a high error protection, which is reasonable and affordable due to its low size.

Replace Table 119 — Syntax of HOAConfig() with:

Table 119 — Syntax of HOAConfig()

Syntax	No. of bits	Mnemonic
HOAConfig() { HoaOrder = escapedValue(3,5,0); NumOfHoaCoeffs = (HoaOrder + 1)^2; IsScreenRelative; UsesNfc; if (UsesNfc) { NfcReferenceDistance; } HOADecoderConfig(); }	3,8 1 1 32	uimbsbf uimbsbf bslbf bslbf
NOTE: HoaOrder = 30 ... 38 are reserved.		

Replace Table 120 — Syntax of HOADecoderConfig() with:

Table 120 — Syntax of HOADecoderConfig()

Syntax	No. of bits	Mnemonic
HOADecoderConfig(numHOATransportChannels) { MinAmbHoaOrder = escapedValue(3,5,0) – 1; MinNumOfCoeffsForAmbHOA = (MinAmbHoaOrder + 1)^2; NumOfAdditionalCoders = numHOATransportChannels – MinNumOfCoeffsForAmbHOA; NumLayers = 1; NumHOAChannelsLayer[0] = numHOATransportChannels; if(SingleLayer == 0){	3,8	uimsbf
}	1	Bslbf

<pre> HOALayerChBits = ceil(log2(NumOfAdditionalCoders)); NumHOAChannelsLayer[0] = codedLayerCh + MinNumOfCoeffsForAmbHOA; remainingCh = numHOATransportChannels – NumHOAChannelsLayer[0]; while (remainingCh>1) { HOALayerChBits = ceil(log2(remainingCh)); NumHOAChannelsLayer[NumLayers] = NumHOAChannelsLayer[NumLayers-1] + codedLayerCh + 1; remainingCh = remainingCh – NumHOAChannelsLayer[NumLayers]; NumLayers++; } if (remainingCh) { NumHOAChannelsLayer[NumLayers] = NumOfAdditionalCoders; NumLayers++; } } </pre>	<pre> HOALayerChBits uimsbf </pre>
<pre> CodedSpatialInterpolationTime; SpatialInterpolationMethod; CodedVVecLength; MaxGainCorrAmpExp; HOAFrameLengthIndicator; </pre>	<pre> 3 uimsbf 1 bslbf 2 uimsbf 3 uimsbf 2 uimsbf </pre>
<pre> if(MinAmbHoaOrder < HoaOrder) { DiffOrderBits = ceil(log2(HoaOrder- MinAmbHoaOrder+1)) MaxHOAOrderToBeTransmitted = DiffOrder + MinAmbHoaOrder; } else { MaxHOAOrderToBeTransmitted = HoaOrder; } MaxNumOfCoeffsToBeTransmitted = (MaxHOAOrderToBeTransmitted + 1)^2; MaxNumAddActiveAmbCoeffs = MaxNumOfCoeffsToBeTransmitted - MinNumOfCoeffsForAmbHOA; VqConfBits = ceil (log2(ceil(log2(NumOfHoaCoeffs+1)))); NumVVecVqElementsBits; if(MinAmbHoaOrder == 1) { UsePhaseShiftDecorr; } </pre>	<pre> DiffOrder uimsbf Bits </pre>
<pre> if(SingleLayer==1) { HOADecoderEnhConfig(); } </pre>	<pre> VqConfBits uimsbf </pre>
<pre> AmbAsignmBits = ceil(log2(MaxNumAddActiveAmbCoeffs)); ActivePredIdsBits = ceil(log2(NumOfHoaCoeffs)); i = 1; while(i * ActivePredIdsBits + ceil(log2(i)) < NumOfHoaCoeffs){ i++; } </pre>	<pre> 1 bslbf </pre>

<pre> NumActivePredIdsBits = ceil(log2(max(1, i - 1))); GainCorrPrevAmpExpBits = ceil(log2(ceil(log2(1.5 * NumOfHoaCoeffs)) + MaxGainCorrAmpExp + 1)); for (i=0; i<NumOfAdditionalCoders; ++i){ AmbCoeffTransitionState[i] = 3; } </pre>
NOTE: MinAmbHoaOrder = 30 ... 37 are reserved. HOAFrameLengthIndicator = 3 is reserved. CodedVVecLength = 3 is reserved.

Add the following tables before Figure 35 — Example for HOAConfig() in subclause 12.2.1 Configuration of HOA elements:

Table AMD3.HLC.1 — Syntax of HOAEnhConfig()

Syntax	No. of Mnemonic bits
<pre> HOAEnhConfig() { HOALayerIdxBits = ceil(log2(NumOfAdditionalCoders+1)); LayerIdx HOADecoderEnhConfig(); } </pre>	HOALayerIdx uimsbf xBits

Table AMD3.HLC.2 — Syntax of HOADecoderEnhConfig()

[illegible]

```

    for ( idx = LastFirstOrderSubBandIdx;
          idx < NumOfParSubbands; idx++) {
        UpmixHoaOrderPerParSubband[idx] = 2;
        MaxNumOfDecoSigs[idx] =
            (UpmixHoaOrderPerParSubband[idx] + 1)^2;
    }
}

```

Table AMD1.1 — Syntax of getSubbandWidths()

Syntax	No. of bits	Mnemonic
<pre> getSubbandWidths(NumberOfSubbands) { totalBwSum = 0; if(NumberOfSubbands > 1) { CodedBwFirstBand bw[0] = CodedBwFirstBand+1; totalBwSum = totalBwSum + bw[0]; if(NumberOfSubbands > 2) { for (nb = 1; nb < NumberOfSubbands-2; nb++) { bw[nb] = bw[nb-1] + bw_diff; totalBwSum = totalBwSum + bw[nb]; } bw[nb] = bw[nb-1] + bw_diff; totalBwSum = totalBwSum + bw[nb]; } } bw[NumberOfSubbands-1] = 64 – totalBwSum; return(bw); } </pre>	1.. 1.. 5	uclbf uclbf uimsbf

Remove Figure 35 — Example for HOAConfig()

Replace Table 121 — Syntax of HOAFrame with:

Table 121 — Syntax of HOAFrame

Syntax	No. of bits	Mnemonic
<pre> HOAFrame() { NumOfDirSigs = 0; NumOfVecSigs = 0; NumOfAddHoaChans = 0; for(lay=0; (lay< NumLayers); ++lay){ NumOfDirSigsPerLayer[lay] = 0; NumOfAddHoaChansPerLayer[lay] = 0; NumOfContAddHoaChans[lay] = 0; NumOfNewAddHoaChans[lay] = 0; } hoalIndependencyFlag; } </pre>	1	bslbf

```

for(i=0; i< NumOfAdditionalCoders; ++i){
    ChannelSideInfoData(i);
    HOAGainCorrectionData(i);
    switch ChannelType[i] {
    case 0:
        DirSigChannelIds[NumOfDirSigs] = i + 1;
        NumOfDirSigs++;
        for(lay=0; (lay< NumLayers); ++lay){
            if( (MinNumOfCoeffsForAmbHOA + i ) <
                NumHOAChannelsLayer[lay]){
                NumOfDirSigsPerLayer[lay]++;
            }
        }
        break;
    case 1:
        VecSigChannelIds[NumOfVecSigs] = i + 1;
        lay = 0;
        while( (MinNumOfCoeffsForAmbHOA + i )
            ≥ NumHOAChannelsLayer[lay]) {
            lay ++;
        }
        VecSigLayerIdx[NumOfVecSigs] = lay;
        NumOfVecSigs++;
        break;
    case 2:
        for(lay=0; (lay< NumLayers); ++lay){
            if( (MinNumOfCoeffsForAmbHOA + i ) <
                NumHOAChannelsLayer[lay]){
                if (AmbCoeffTransitionState[i] == 0) {
                    ContAddHoaCoeff[lay]
                        [NumOfContAddHoaChans[lay]]
                        = AmbCoeffIdx[i];
                    NumOfContAddHoaChans[lay]++;
                }
            }else{
                if(AmbCoeffTransitionState[i] == 1) {
                    NewAddHoaCoeff[lay]
                        [NumOfNewAddHoaChans]
                        = AmbCoeffIdx[i];
                    NumOfNewAddHoaChans[lay]++;
                }
            }

            AddHoaCoeffPerLayer[lay]
                [NumOfAddHoaChans]
                = AmbCoeffIdx[i];
            NumOfAddHoaChansPerLayer[lay]++;
        }

        AddHoaCoeff[NumOfAddHoaChans] = AmbCoeffIdx[i];
        NumOfAddHoaChans++;
        break;
    }
}

for ( i= NumOfAdditionalCoders;
    i< NumHOATransportChannels; ++i){
    HOAGainCorrectionData(i);
}

```

<pre> for(i=0; i< NumOfVecSigs; ++i){ VVectorData (VecSigChannelIds(i)); } if(SingleLayer==1) { HOAEnhFrame(); } </pre>
NOTE: the encoder shall set hoalIndependencyFlag to 1 if usacIndependencyFlag (see mpeg3daFrame() in Table 26) is set to 1. NOTE: If SingleLayer == 1 set NumLayers = 1.

Add the following table into Section 12.2.2 after Table 121

Table AMD3.HLC.3 — Syntax of HOAEnhFrame

Syntax	No. of bits	Mnemonic
<pre> HOAEnhFrame() { if(((SingleLayer==1) & (NumOfDirSigs > 0)) ((SingleLayer==0) & (NumOfDirSigsPerLayer[lay] > 0))) { HOAPredictionInfo() } if(NumOfPredSubbands > 0) { HOADirectionalPredictionInfo(); } if(NumOfParSubbands > 0) { HOAParInfo(); } } </pre>		
Note: lay is the index of the currently active HOA enhancement layer If SingleLayer==0 then use NumOfPredSubbands and NumOfParSubbands from the HOADecoderEnhConfig of the corresponding layer with index lay		

Replace Table 122 — Syntax of ChannelSideInfoData(i) with:

Table 122 — Syntax of ChannelSideInfoData(i)

Syntax	No. of bits	Mnemonic
<pre> ChannelSideInfoData(i) { ChannelType[i] switch ChannelType[i] { case 0: ActiveDirIds[i]; break; </pre>		
	2	uimsbf
	10	uimsbf

case 1:		
if(hoaIndependencyFlag){		
if(CodedVVecLength==1){		
NewChannelTypeOne(k)[i];	1	bslbf
}		
NbitsQ(k)[i]	4	uimsbf
if (NbitsQ(k)[i] == 4) {		
CodebkIdx(k)[i];	3	uimsbf
NumVvecIndices(k)[i]++;	NumVVecV qElements Bits	uimsbf
}		
elseif (NbitsQ(k)[i] >= 6) {		
PFlag(k)[i] = 0;		
CbFlag(k)[i];	1	bslbf
}		
}		
else{		
if(CodedVVecLength==1){		
NewChannelTypeOne(k)[i] = (1!=ChannelType(k-1)[i]);		
}		
bA;	1	bslbf
bB;	1	bslbf
if ((bA + bB) == 0) {		
NbitsQ(k)[i] = NbitsQ(k-1)[i];		
PFlag(k)[i] = PFlag(k-1)[i];		
CbFlag(k)[i] = CbFlag(k-1)[i];		
CodebkIdx(k)[i] = CodebkIdx(k-1)[i];		
NumVvecIndices(k)[i] =		
NumVvecIndices(k-1)[i];		
}		
else{		
NbitsQ(k)[i] = (8*bA)+(4*bB)+uintC;	2	uimsbf
if (NbitsQ(k)[i] == 4) {		
CodebkIdx(k)[i];	3	uimsbf
NumVvecIndices(k)[i]++;	NumVVecV qElements Bits	uimsbf
}		
elseif (NbitsQ(k)[i] >= 6) {		
PFlag(k)[i];	1	bslbf
CbFlag(k)[i];	1	bslbf
}		
}		
}		
break;		
case 2:		
AddAmbHoaInfoChannel(i);		
break;		
default:		
}		
}		
NOTE CodebkIdx = 4 ... 6 are reserved.		

Replace Table 126 — Syntax of *VVectorData()* with:

Syntax	No. of bits	Mnemonic
<pre> VVectorData(i) { if (CodedVVecLength == 1) { VVecLengthUsed = VVecLength[i]; VVecCoeffIdxUsed = VVecCoeffIdx[i]; } else { VVecLengthUsed = VVecLength; VVecCoeffIdxUsed = VVecCoeffIdx; } if (NbitsQ(k)[i] == 4) { if (NumVVecIndices(k)[i] == 1) { VvecIdx[0] = VvecIdx + 1; WeightVal[0] = ((SgnVal*2)-1); } else { WeightIdx; for (j=0; j< NumVVecIndices(k)[i]; ++j) { VvecIdx[j] = VvecIdx + 1; if (j<8) { WeightVal[j] = ((SgnVal*2)-1) * WeightValCdbk[WeightIdx][j]; } else { WeightVal[j] = ((SgnVal*2)-1) * WeightValCdbk[WeightIdx][6+j%2]; } } } } else if (NbitsQ(k)[i] == 5) { for (m=0; m< VVecLengthUsed; ++m){ aVal[i][m] = (VecVal / 128.0) - 1.0; } } else if (NbitsQ(k)[i] >= 6) { for (m=0; m< VVecLengthUsed; ++m){ huffIdx = <i>huffSelect</i>(VVecCoeffIdxUsed[m], PFlag[i], CbFlag[i]); cid = <i>huffDecode</i>(NbitsQ[i], huffIdx, huffVal); aVal[i][m] = 0.0; if (cid > 0) { aVal[i][m] = sgn = (sgnVal * 2) - 1; if (cid > 1) { aVal[i][m] = sgn * (2.0^(cid - 1)) + intAddVal; } } } } } </pre>	nbitsIdx 1 8 nbitsIdx 1 1 8 dynamic 1 cid-1	uimsbf uimsbf uimsbf uimsbf uimsbf uimsbf uimsbf huffDecode bslbf uimsbf
NOTE: See 12.4.1.10.X for computation of VVecLength and nbitsIdx		

Include following changes in Table 127 Syntax of HOAPredictionInfo(), which is defined in Section 12.2.2 (changes highlighted in grey)

Table 127 — Syntax of HOAPredictionInfo()

Syntax	No. of bits	Mnemonic
<pre> HOAPredictionInfo() { PredIdsBits = ceil(log2(NumOfDirSigsPerLayer[lay] + 1)); if(PSPredictionActive){ NumActivePred = 0; if(KindOfCodedPredIds){ NumActivePred = NumActivePredIds + 1; i=0; while(i < NumActivePred){ PredIds[i] = PredIds[i] + 1; i++; } } else{ for (i=0; i<(HoaOrder +1)^2; i++) { if(ActivePred[i]) { NumActivePred ++; } } } NumOfGains=0; for (i=0; i<NumActivePred * MaxNoOfDirSigsForPrediction; i++) { if (PredDirSigIds[i] > 0) { PredDirSigIds[i] = DirSigChannelIds[PredDirSigIds[i] - 1]; NumOfGains++; } } for (i=0; i< NumOfGains; i++) { bsPredGains PredGains[i] = bsPredGains – 2^(NoOfBitsPerScalefactor-1) } } } </pre>	1 1 NumActivePredIdsBits ActivePredIdsBits 1 PredIdsBits NoOfBitsPerScalefactor	bslbf bslbf uimbsf uimbsf bslbf uimbsf uimbsf
Note: lay is the index of the currently active HOA enhancement layer. In case of SingleLayer==1 set lay to zero.		

Add the following tables before Figure 36 — Two examples for HOAFrame()

Table AMD1.2 — Syntax of HOADirectionalPredictionInfo()[illegible]

```
PredDirGridIdx[band][dir] =  
    GlobalPredDirsIds[RelDirGridIdx];  
for ( hoalidx = 0;  
     hoalidx < MinNumOfCoeffsForAmbHOA;  
     hoalidx++ ) {  
    readDirPredDiffValues (band, dir, hoalidx,  
        UseHuffmanCodingDiffAbs,  
        UseHuffmanCodingDiffAngle,  
        FirstSBRSubbandIdx);  
}  
for ( idx = 0;  
     idx < NumOfAddHoaChansPerLayer[layers];  
     idx++ ) {  
    readDirPredDiffValues (band, dir,  
        SortedAddHoaCoeff[idx] - 1,  
        UseHuffmanCodingDiffAbs,  
        UseHuffmanCodingDiffAngle,  
        FirstSBRSubbandIdx);  
}  
  
}  
  
}  
  
}  
  
}
```

Note: lay is the index of the currently active HOA enhancement layer. In case of SingleLayer==1 set lay to zero.

Table AMD1.3 — Syntax for readDirPredDiffValues()

Syntax	No. of bits	Mnemonic
<pre> readDirPredDiffValues(band, dir, hoalidx, UseHuffmanCodingDiffAbs, UseHuffmanCodingDiffAngle, FirstSBRSubbandIdx) { if(UseHuffmanCodingDiffAbs) { if(band < FirstSBRSubbandIdx) { DecodedMagDiff[band][dir][hoalidx] = HuffmanMagDiffNoSbr[HuffmanCodedMagDiff]; } else { DecodedMagDiff[band][dir][hoalidx] = HuffmanMagDiffSbr[HuffmanCodedRealMagDiff]; } if(DecodedMagDiff[band][dir][hoalidx] ≤ -8){ DecodedMagDiff[band][dir][hoalidx] = -8 - runLengthCodedVal; { else if (DecodedMagDiff[band][dir][hoalidx] ≥ 9){ DecodedMagDiff[band][dir][hoalidx] = 9 + runLengthCodedVal; } } } } else { DecodedMagDiff[band][dir][hoalidx] = CodedMagDiff -7; if(DecodedMagDiff[band][dir][hoalidx] ≤ -7){ DecodedMagDiff[band][dir][hoalidx] = -7 - runLengthCodedVal; } } } </pre>	1..10 1..12 1.. 1.. 4 1..	vlclbf vlclbf uclbf uclbf uimsbf uclbf

<pre> else if (DecodedMagDiff[band][dir][hoaldx] ≥ 8){ DecodedMagDiff[band][dir][hoaldx] = 8 + runLengthCodedVal; } } if(band < FirstSBRSubbandIdx) { if(UseHuffmanCodingDiffAngle) { DecodedAngleDiff[band][dir][hoaldx] = DecTableAngleDiff[HuffCodedAngleDiff]; } else { DecodedAngleDiff[band][dir][hoaldx] = DecTableAngleDiff[CodedAngleDiff]; } } } </pre>	1..	uclbf
<pre> DecodedAngleDiff[band][dir][hoaldx] = DecTableAngleDiff[HuffCodedAngleDiff]; </pre>	1..7	vlclbf
<pre> DecodedAngleDiff[band][dir][hoaldx] = DecTableAngleDiff[CodedAngleDiff]; </pre>	4	uimsbf

Table AMD1.4 — Syntax of HOAParInfo()

Syntax	No. of bits	Mnemonic
<pre> HOAParInfo() { if (UsePar) { for (band = 0; band < NumOfParSubbands; band++) { for(n = 0; n < MaxNumOfDecoSigs[band]; n++) { for(m = 0; m < MaxNumOfDecoSigs[band]; m++) { DecodedParMagDiff [band][n][m] = 0; DecodedParAngleDiff [band][n][m] = 0; } } if (!hoalIndependencyFlag) { KeepPreviousParMatrixFlag[band]; } else{ KeepPreviousParMatrixFlag[band] = 0; } if (!KeepPreviousParMatrixFlag[band]) { ParDecorrSigsSelectionTableIdx[band] if (UpmixHoaOrderPerParSubband[band] == 2) { NumOfDecorrSigsPerParSubband = NumOfDecorrSigsPerParSubbandTable[ParDecorrSigsSelectionTableIdx[band]] ParSelectedDecorrSigsIdxMatrix = ParSelectedDecorrSigsIdxMatrixTable[ParDecorrSigsSelectionTableIdx[band]] } else{ NumOfDecorrSigsPerParSubband = NumOfDecorrSigsPerFirstOrderPar SubbandTable[ParDecorrSigsSelectionTableIdx[band]] ParSelectedDecorrSigsIdxMatrix = ParFirstOrderSelectedDecorr SigsIdxMatrixTable [ParDecorrSigsSelectionTableIdx[band]] } } } } } </pre>	1	bslbf
<pre> KeepPreviousParMatrixFlag[band]; </pre>	1	bslbf
<pre> ParDecorrSigsSelectionTableIdx[band] </pre>	2	uimsbf

if (UseReducedNoOfUpmixSigs) { for(n = 0; n < MaxNumOfDecoSigs[band]; n++) { UseParUpmixSig [band][n]; } } else { for(n = 0; n < MaxNumOfDecoSigs[band]; n++) { UseParUpmixSig [band][n] = 1; } }	1	bslbf
UseParHuffmanCodingDiffAbs;	1	bslbf
if(! UseRealCoeffsPerParSubband [band]) { UseParHuffmanCodingDiffAngle; }	1	bslbf
for(n = 0; n < MaxNumOfDecoSigs[band]; n++) { if(UseParUpmixSig [band][n]) { for(m = 0; m < NumOfDecorrSigsPerParSubband ; m++) { c = ParSelectedDecorrSigsIdxMatrix [n][m]; readParDiffValues(band, n, c, UseParHuffmanCodingDiffAbs, UseParHuffmanCodingDiffAngle); } } } }		

Table AMD1.5 — Syntax for readParDiffValues()

Syntax	No. of bits	Mnemonic
<pre> readParDiffValues (band, idx, decoldx, UseParHuffmanCodingDiffAbs, UseParHuffmanCodingDiffAngle) { if(UseParHuffmanCodingDiffAbs) { if(UseRealCoeffsPerParSubband[band]) { DecodedParMagDiff[band][idx][decoldx] = HuffmanMagDiffSbr[HuffmanCodedParMagDiff]; } else { DecodedParMagDiff[band][idx][decoldx] = HuffmanMagDiffNoSbr[HuffmanCodedRealParMagDiff]; } if (DecodedParMagDiff [band][dir][hoalidx] ≤ -8) DecodedParMagDiff[band][idx][decoldx] = -8 - runLengthCodedVal; } else if (DecodedParMagDiff [band][dir][hoalidx] ≥ 9) DecodedParMagDiff[band][idx][decoldx] = 9 + runLengthCodedVal; } } else { DecodedParMagDiff[band][idx][decoldx] = CodedParMagDiff - 7; } } </pre>	1..10 1..12 1.. 1.. 4	vlclbf vlclbf uclbf uclbf uimbsf

if (DecodedParMagDiff [band][dir][hoaldx] ≤ -7) DecodedParMagDiff[band][idx][decoldx] = -7 - runLengthCodedVal ;	1..	uclbf
else if (DecodedParMagDiff [band][dir][hoaldx] ≥ 8) DecodedParMagDiff[band][idx][decoldx] = 8 + runLengthCodedVal ;	1..	uclbf
if(!UseRealCoeffsPerParSubband[band]){ if(UseParHuffmanCodingDiffAngle) { DecodedParAngleDiff[band][idx][decoldx] = DecTableAngleDiff[HuffCodedParAngleDiff]; } else { DecodedParAngleDiff[band][idx][decoldx] = DecTableAngleDiff[CodedParAngleDiff]; } } }	1..7	vlclbf
	4	uimsbf

Replace "Table 128 – SingleLayer definition" with the following table and add the additional definitions of elements following the new Table 128 :

Table 128 — SingleLayer definition

Value	Meaning
0	/* reserved */
1	HOA signal is provided in a single layer

With:

Table 128 — SingleLayer definition

Value	Meaning
0	HOA signal is provided in multiple layers; enables the signaling of the distribution of the HOA transport channels into the different layers
1	HOA signal is provided in a single layer

codedLayerCh

This element indicates for the first (i.e. base) layer the number of included transport signals, which is given by **codedLayerCh** + MinNumOfCoeffsForAmbHOA. For the higher (i.e. enhancement) layers, this element indicates the number of additional signals included into an enhancement layer compared to the next lower layer, which is given by **codedLayerCh** + 1.

HOALayerChBits

This element indicates the number of bits for reading **codedLayerCh**.

NumLayers

This element indicates (after the reading of the HOADecoderConfig()) the total number of layers within the bit stream.

NumHOACannelsLayer

This element is an array consisting of NumLayers elements, of which the i-th element indicates the number of transport signals included in all layers up to the i-th layer.

HOAEnhConfig()	This payload contains the HOA enhancement configuration data of an HOA enhancement layer in the HOA Layered Coding Mode.
HOALayerIdxBits	This element indicates the number of bits used for signaling the element LayerIdx.
LayerIdx	This element indicates the index of the HOA enhancement layer payload, where the index is defined in the range from zero to NumLayers-1.
HOADecoderEnhConfig()	This payload contains all configuration elements for the HOA enhancement layer payloads.

In subclause 12.3.1 Definitions of HOA Config replace:

NumOfHoaCoeffs	This element determines the number of HOA coefficients of the coded HOA representation, which is equal to the number of HOA coefficients to be reconstructed.
----------------	---

With:

NumOfHoaCoeffs	This element determines the number of HOA coefficients of the coded HOA representation, which is equal to the number of HOA coefficients to be reconstructed.
----------------	---

IsScreenRelative	This element indicates if the HOA representation shall be rendered with respect to the reproduction screen size as described in clause 18.X.
------------------	--

In subclause 12.3.1 Definitions of HOA Config replace:

MaxNumAddActiveAmbCoeffs	This element determines the maximum number of additional HOA channels that can be used additionally for the coding of the ambient HOA representation.
--------------------------	---

With:

MaxHOAOrderToBeTransmitted	This element indicates the maximum HOA order of the additional ambient HOA coefficients to be transmitted.
----------------------------	--

MaxNumOfCoeffsToBeTransmitted	This element indicates the maximum number of HOA coefficients to be transmitted, computed depending on MaxHOAOrderToBeTransmitted.
-------------------------------	--

MaxNumAddActiveAmbCoeffs	This element signals the maximum index for the signaling of additional ambient HOA coefficients.
--------------------------	--

VqConfBits	This element indicates the number of bits necessary to signal the element NumVVecVqElementsBits
------------	--

NumVVecVqElementsBits	This element indicates the number of bits used to signal the element NumVvecIndices in ChannelSideInfoData()
------------------------------	---

UsePhaseShiftDecorr	This element signals the usage of the phase shift de-correlation in the ambience synthesis, where true means it is used.
----------------------------	--

PredSubbandsIdx	This element signals the table index for the sub-band configuration of the Sub-band Directional Signals Synthesis
------------------------	---

NumOfPredSubbands	This element contains the number of sub-band groups used for the Sub-band Directional Signals Synthesis
NumOfPredSubbandsTable	This table contains the number of sub-band groups for each PredSubbandsIdx for the Sub-band Directional Signals Synthesis
PredSubbandWidths[idx]	This array of NumOfPredSubbands elements contains the number of QMF sub-bands per sub-band group of the Sub-band Directional Signals Synthesis.
PredSubbandWidthTable	This table contains the bandwidths of the sub-band groups for each table index PredSubbandsIdx for the Sub-band Directional Signals Synthesis

Table AMD1.6 — Directional Prediction Subbands Table

PredSubbandsIdx	NumOfPredSubbandsTable	PredSubbandWidthTable
0	0	[]
1	10	[1, 1, 1, 2, 2, 2, 3, 6, 11, 35]
2	20	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 4, 5, 7, 11, 18]

getSubbandWidths()	This function reads a flexible sub-band group configuration.
FirstSBRSubbandIdxBits	This element determines the number of bits for reading FirstSBRSubbandIdx .
FirstSBRSubbandIdx	Indicates the first sub-band group of the Directional Signals Synthesis where the samples are reconstructed by the SBR tool. To deactivate the special SBR processing in the Sub-band Directional Signals Synthesis set the value of this element to NumOfPredSubbands + 1.
MaxNumOfPredDirsLog2	This element signals the logarithm to the base of two from the maximum number of signals that are predicted in the Sub-band Directional Signals Synthesis.
MaxNumOfPredDirs	This element signals the maximum number of signals that are predicted in the Sub-band Directional Signals Synthesis.
MaxNumOfPredDirsPerBand	This element contains the maximum number of predicted signals per sub-band group of the Sub-band Directional Signals Synthesis.
DirGridTableIdx	This index determines the grid of potential directions of directional sub-band signals created at the Sub-band Directional Signals Synthesis.
NumOfBitsPerDirIdx	The element determines the number of bits for signaling the index of a virtual loudspeaker position index.
NumOfGridPointsTable	This table determines the number of potential directions of directional sub-band signals created at the Sub-band Directional Signals Synthesis.
NumOfBitsPerDirIdxTable	This table determines the number of bits used to code each single potential direction of a directional sub-band signal created at the Sub-band Directional Signals Synthesis.

Table AMD1.7 — Quantized Direction Index Table

DirGridTableIdx	NumOfGridPointsTable	NumOfBitsPerDirIdxTable
0	256	8
1	484	9
2	900	10
3	reserved	reserved

ParSubbandTableIdx	This element signals the table index for the sub-band configuration of the Parametric Ambience Replication decoder
NumOfParSubbands	This element signals the number of sub-band groups used by the Parametric Ambience Replication decoder
NumOfParSubbandsTable	This table contains the number of sub-band groups for each PAR sub-band group table index ParSubbandTableIdx
ParSubbandWidths	This array of NumOfParSubbands elements contains the number of QMF sub-bands per sub-band group of the Parametric Ambience Replication decoder.
ParSubbandWidthTable	This table contains the bandwidths of the sub-band groups for each table index ParSubbandTableIdx for the PAR decoder

Table AMD1.8 — PAR Subbands Table

ParSubbandTableIdx	NumOfParSubbandsTable	ParSubbandWidthTable
0	0	[]
1	4	[1, 1, 22, 40]
2	8	[1, 1, 1, 2, 2, 5, 10, 42]

LastFirstOrderSubbandIdxBits	This element determines the number of bits required for reading LastFirstOrderSubbandIdx
LastFirstOrderSubbandIdx	This element indicates the index of the last PAR sub-band group that uses an HOA order of one.
UseRealCoeffsPerParSubband[idx]	This Boolean array indicates for each PAR sub-band group if the mixing matrix consists of real-valued non-negative (true) or complex valued elements matrix (false).
UpmixHoaOrderPerParSubband[idx]	This array contains the HOA order used in each PAR sub-band group
MaxNumOfDecoSigs[idx]	This array contains the maximum number of de-correlated signals per PAR sub-band group.

Add the following subclause after subclause 12.3.1 *Definitions of HOA Config*

12.3.x Syntax of getSubbandBandwidths()

This function reads the bandwidth for NumberOfSubbands sub-band groups.

CodedBwFirstBand	This element signals the bandwidth of the first sub-band group by the number of QMF sub-bands
NumberOfSubbands	This element contains the total number of sub-band groups

bw[idx]	This array holds the bandwidth of each sub-band group in the number of QMF sub-bands
bw_diff	This element contains the differentially coded bandwidth of a sub-band group

Replace the subclause 12.3.2.1 HOAFrame() with the following subclause

12.3.2.1 HOAFrame()

The HOAFrame() holds the information that is required to decode the L samples of a HOA frame of order $HoaOrder$.

HOAFrame()	This block of data contains the data required to decode one frame of L samples of the coded HOA representation.
NumOfDirSigs	This elements determines the number of active directional signals in the current HOAFrame().
NumOfVecSigs	This element determines the number of active vector-based signals in the current HOAFrame().
NumOfAddHoaChans	This element determines the total number of additional ambient HOA channels in the current HOAFrame().
NumOfDirSigsPerLayer[lay]	This elements determines the number of active directional signals in the current HOAFrame() that are actually used in the HOA enhancement layer <i>lay</i> .
NumOfAddHoaChansPerLayer[lay]	This element signals the total number of additional ambient HOA coefficients actually used in the HOA enhancement layer <i>lay</i> .
NumOfContAddHoaChans[lay]	This elements determines the number of continuous additional ambient HOA coefficients in the current HOAFrame() that are actually used in the HOA enhancement layer <i>lay</i> .
NumOfNewAddHoaChans[lay]	This element determines the number of newly introduced additional ambient HOA channels in the current HOAFrame() that are actually used in the HOA enhancement layer <i>lay</i> .
hoaindependencyFlag	This flag signals that the current frame is an independent frame that can be decoded without having knowledge about the previous frame. Otherwise this flag is equal to the <code>usaindependencyFlag</code> .
ChannelSideInfoData(i)	This payload holds the side information for the i -th of the <code>NumOfAdditionalCoders</code> channels of flexible <code>ChannelType</code> .
HOAGainCorrectionData(i)	This payload contains data for the inverse gain correction of channel i .
DirSigChannelIds[NumOfDirSigs]	This element stores the channel index of each active directional signal of the current frame.
VecSigChannelIds[NumOfVecSigs]	This element stores the channel index of each active vector-based signal of the current frame.

VecSigLayerIdx[i] This element indicates for the i-th vector based channel element of the current frame the index of the enhancement layer that transmits the corresponding vector-based signal.

ContAddHoaCoeff [lay][NumOfContAddHoaChans]

This 2D array contains the HOA coefficient indices for each continuous additional ambient HOA coefficient actually used in the HOA enhancement layer *lay*. For these HOA coefficients the AmbCoeffTransitionState word is of value 0 in the current frame.

NewAddHoaCoeff [lay][NumOfContAddHoaChans]

This 2D array contains the HOA coefficient indices for each additional ambient HOA coefficient, which was not present in the preceding frame used in the HOA enhancement layer *lay*.

AddHoaCoeffPerLayer[lay] This array contains the HOA coefficient indices for each additional ambient HOA coefficient actually used in the HOA enhancement layer *lay*.

AddHoaCoeff[lay] This element stores the HOA coefficient number of each additional ambient HOA channel in the current frame.

HOAEnhFrame() This payload contains the frame data for the HOA enhancement payloads HOAPredictionInfo(), HOADirectionalPredictionInfo() and HOAParInfo().

Add the subclause 12.3.2.X after 12.3.2.1 HOAFrame()

12.3.2.X HOAEnhFrame()

HOAPredictionInfo()

This payload contains data for the prediction of dominant sound sources from the active directional signals of the current frame.

HOADirectionalPredictionInfo This payload contains data for the Sub-band Directional Signals Synthesis.

HOAParInfo This payload contains data for the Parametric Ambience Replication.

In subclause 12.3.2.2 ChannelSideInfoData(i) add after CodebkIdx[i]

NumVvecIndices(k)[i] The number of vectors used to dequantize a vector-quantized V-vector.

In subclause 12.3.2.2 ChannelSideInfoData(i) add before PFlag[i]

NewChannelTypeOne[i] This flag indicates if in the previous frame (k-1) the transport channel was not initialized as a Vector-based Signal.

In subclause 12.3.2.5 *VVectorData(VecSigChannelIds(i))* remove:

NumVvecIndices The number of vectors used to dequantize a vector-quantized V-vector.

Add the following subclauses after 12.3.2.6 *HOAPredictionInfo()*:

12.3.2.x HOADirectionalPredictionInfo

This Payload contains data for the Directional Signals Synthesis.

UseDirectionalPrediction	This flag indicates if the Directional Signals Synthesis is performed in the current frame.
KeepPreviousGlobalPredDirsFlag	This flag indicates that the elements NumOfGlobalPredDirs, NumBitsForRelDirGridIdx and GlobalPredDirIds read from the previous HOADirectionalPredictionInfo payload are used.
NumOfGlobalPredDirs	This element determines the available number of global, broadband directions.
NumBitsForRelDirGridIdx	This element indicates the number of bit required for reading GlobalPredDirIds .
GlobalPredDirIds	This array contains the indices for the NumOfGlobalPredDirs directions, where each index belongs to a direction from the grid of positions selected by DirGridTableIdx .
SortedAddHoaCoeff	This array contains the HOA coefficients indices from AddHoaCoeff sorted ascendingly.
KeepPreviousDirPredMatrixFlag[band]	This Boolean is true if the signal prediction matrix is kept from the previous frame and false if the mixing matrix is update by new magnitude and angle differences.
UseHuffmanCodingDiffMag	This Boolean element is true if the magnitude differences are encoded by Huffman coding.
UseHuffmanCodingDiffAngle	This Boolean element is true if the angle differences are encoded by Huffman coding.
DirIsActive	This Boolean element is true if the current direction of the current sub-band group has been transmitted.
RelDirGridIdx	This element holds the relative index of the current direction index stored in GlobalPredDirIds
PredDirGridIdx[band][dir]	This 2D array contains the direction indices belonging to the direction grid selected by DirGridTableIdx for all active directions of each sub-band group
readDirPredDiffValues()	This function reads the coded magnitude and angle differences.

12.3.2.x.1 Syntax for readDirPredDiffValues()

This function reads the magnitude and phase differences for the Directional Signal Synthesis.

HuffCodedMagDiff	This element signals the Huffman coded magnitude difference using Table F.x Huffman Table for Decoding HuffmanMagDiffNoSbr
DecodedMagDiff[band][dir][idx]	This 3D array holds the decoded magnitude difference from Table F.x Huffman Table for Decoding HuffmanMagDiffSbr for the prediction of each directional signal with the index dir of a sub-band group with the index band from the HOA coefficient with the index idx
runLengthCodedVal	This run length code determines magnitude difference if a corresponding escape word has been signaled.
HuffCodedAngleDiff	This element signals the Huffman coded angle difference using Table F.x Huffman Table for DecTableAngleDiff
DecodedAngleDiff[band][dir][hoaldx]	This 3D array holds the decoded angle difference from Table F.x Huffman Table for DecTableAngleDiff for the prediction of each directional signal with the index dir of a sub-band group with the index band from the HOA coefficient with the index idx
HuffCodedSbrMagDiff	This element signals the Huffman coded magnitude difference using Table F.x Huffman Table for Decoding HuffmanMagDiffSbr .
CodedMagDiff	This element holds the magnitude difference index quantized with 4 plain bits.
CodedAngleDiff	This element holds the angle difference index quantized with 4 plain bits.

12.3.2.x HOAParInfo

This Payload contains data for the PAR decoder.

UsePar	This Boolean is true if the PAR decoding is performed in the current frame.
KeepPreviousParMatrixFlag[idx]	This Boolean is true if the mixing matrix is kept from the previous frame and false if the mixing matrix is update by new magnitude and angle differences.
ParDecorrSigsSelectionTableIdx[idx]	This table index determines the number and the corresponding indices of the de-correlated signals that are used to create one up-mix signal.
NumOfDecorrSigsPerParSubband	This element indicates the number of de-correlated signals that are used to create one up-mix signal, which is obtained for UpmixHoaOrderPerParSubband[idx] of two from Table F.x Table for ParDecorrSigsSelectionTableIdx referring to NumOfDecorrSigsPerParSubbandTable and ParSelectedDecorrSigsIdxMatrixTable and for UpmixHoaOrderPerParSubband[idx] of one from Table F.x Table for ParDecorrSigsSelectionTableIdx referring to NumOfDecorrSigsPerFirstOrderParSubbandTable and ParFirstOrderSelectedDecorrSigsIdxMatrixTable exploiting the index ParDecorrSigsSelectionTableIdx[idx]
ParSelectedDecorrSigsIdxMatrix[sig][idx]	This matrix contains the NumOfDecorrSigsPerParSubband indices of the de-correlated signals that are used to create the up-mix signal with the index sig, where the matrix is obtained for UpmixHoaOrderPerParSubband[idx] of two from Table F.x Table for ParDecorrSigsSelectionTableIdx referring to NumOfDecorrSigsPerParSubbandTable and ParSelectedDecorrSigsIdxMatrixTable and for

UpmixHoaOrderPerParSubband[idx] of one from **Table F.x Table for ParDecorrSigsSelectionTableIdx referring to ParFirstOrderPermlDxVectorTable** exploiting the index ParDecorrSigsSelectionTableIdx[idx].

UseReducedNoOfUpmixSigs This elements signals that not all up-mix signals are created.

UseParUpmixSig[idx][n] For each PAR sub-band group of index idx the elements of the Boolean matrix are true if the up-mix signal with an index n is created or false if it is not created.

UseParHuffmanCodingDiffAbs This Boolean indicates by true that the magnitude differences are Huffman coded.

UseParHuffmanCodingDiffAngle This Boolean indicates by true that the angle differences are Huffman coded.

The index is equal to the index of the de-correlated signal used to create the current up-mix signal. It is obtained from **Table F.x Table for ParDecorrSigsSelectionTableIdx referring to NumOfDecorrSigsPerParSubbandTable and ParSelectedDecorrSigsIdxMatrixTable** for the current value of ParDecorrSigsSelectionTableIdx.

12.3.2.x.1 readParDiffValues()

This function reads magnitude and phase differences of the mixing matrices that are used by the PAR decoder.

HuffCodedParMagDiff This element signals the Huffman coded magnitude difference using **Table F.x Huffman Table for Decoding ParHuffmanMagDiffNoSbr**

runLengthCodedVal This run length code determines magnitude difference if a corresponding escape word has been signaled.

DecodedParMagDiff[band][idx][decoldx] This 3D array holds the decoded magnitude differences for the prediction of each virtual loudspeaker signal with the index idx of a sub-band group with the index band from the de-correlated signal with the index decoldx

HuffCodedParAngleDiff This element signals the Huffman coded angle difference using **Table F.x Huffman Table for ParDecTableAngleDiff**

DecodedParAngleDiff[band][idx][decoldx] This 3D array holds the decoded angle differences for the prediction of each virtual loudspeaker signal with the index idx of a sub-band group with the index band from the de-correlated signal with the index idx

HuffmanCodedRealParMagDiff This element signals the Huffman coded magnitude difference using **Table F.x Huffman Table for Decoding ParHuffmanMagDiffSbr**

CodedParMagDiff This element holds the coded magnitude difference index quantized with 4 plain bits.

CodedParAngleDiff This element holds the coded angle difference index quantized with 4 plain bits.

In subclause 12.4.1.2 Global Parameter replace:

$D_{\text{PRED}} = \text{MaxNoOfDirSigsForPrediction}$ Maximum number of directional signals used for the prediction of dominant sound sources

With:

$D_{\text{PRED}} = \text{MaxNoOfDirSigsForPrediction}$ Maximum number of directional signals used for the prediction of dominant sound sources

$D_{\text{SB}} = \text{MaxNumOfPredDirsPerBand}$ Maximum number of directions per sub-band for Sub-band Directional Signals Synthesis

$D_{\text{MAX}} = \text{MaxNumOfPredDirs};$ Maximum number of different directions over all sub-bands for Sub-band Directional Signals Synthesis

In subclause 12.4.1.2 Global Parameter replace:

r_{max} Determines the maximum speaker distance of the listening setup.

With:

r_{max} Determines the maximum speaker distance of the listening setup. In case the actual loudspeaker distances are unknown (hasLoudspeakerDistance == 0) r_{max} is set to infinity ($r_{\text{max}} = \infty$).

Add the following subclauses after 12.4.1.2 Global Parameter

12.4.1.2.1 Upper and lower bounds of sub-band groups for Sub-band Directional Signals Synthesis

$B = \text{NumOfPredSubbands};$

$\mathcal{L}(1) = 1;$

```
for (b=0; b < NumOfPredSubbands - 1; b++)
{
     $\mathcal{U}(b+1) = \mathcal{L}(b) + \text{PredSubbandWidths}[b] - 1;$ 
     $\mathcal{L}(b+1) = \mathcal{U}(b+1) + 1;$ 
}
 $\mathcal{U}(B) = \mathcal{L}(B-1) + \text{PredSubbandWidths}[B-1] - 1;$ 
```

12.4.1.2.2 Upper and lower bounds of sub-band groups for PAR