INTERNATIONAL STANDARD

ISO/IEC 23008-3

Second edition
2019-02-20

AMENDMENT 1
2019-06

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

Part 3:
3D audio

AMENDMENT 1: Audio metadata enhancements

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

Partie 3: Audio 3D

AMENDEMENT 1: Améliorations de la prise en charge des métadonnées audio

Reference number
ISO/IEC 23008-3:2019/Amd.1:2019(E)

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.

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 ISO documents 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 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 of the voluntary nature of standards, 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: www.iso.org/iso/foreword.html.

This document was prepared by Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

A list of all parts in the ISO/IEC 23008 series can be found on the ISO website.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

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

Part 3:
3D audio

AMENDMENT 1: Audio metadata enhancements

5.2.2.1 General configuration syntax

In subclause 5.2.2.1 replace Table 14 with:

Table 14 — Syntax of Signals3d()

Syntax	No. of bits	Mnemonic
Signals3d()		
{		
numAudioChannels = 0;		
numAudioObjects = 0;		
numSAOCTransportChannels = 0;		
numHOATransportChannels = 0;		
bsNumSignalGroups;	5	**uimsbf**
for (grp = 0; grp < bsNumSignalGroups + 1 ; grp++) {		
signal_groupID[grp] = grp;		
differsFromReferenceLayout[grp] = 0;		
signalGroupType[grp];	3	**bslbf**
bsNumberOfSignals[grp] = escapedValue(5, 8, 16);		
if (SignalGroupType[grp] == SignalGroupTypeChannels) {		
numAudioChannels += bsNumberOfSignals[grp] + 1;		
differsFromReferenceLayout[grp];	1	**bslbf**
if(differsFromReferenceLayout[grp]) {		
audioChannelLayout[grp] = SpeakerConfig3d();		
}		
else {		
audioChannelLayout[grp] = referenceLayout;		
}		
}		
if (SignalGroupType[grp] == SignalGroupTypeObject) {		
numAudioObjects += bsNumberOfSignals[grp] + 1;		
}		
if (SignalGroupType[grp] == SignalGroupTypeSAOC) {		
numSAOCTransportChannels += bsNumberOfSignals[grp] + 1;		

Table 14 *(continued)*

Syntax	No. of bits	Mnemonic
saocDmxLayoutPresent;	**1**	**bslbf**
if (saocDmxLayoutPresent == 1) {		
saocDmxChannelLayout = SpeakerConfig3d();		
}		
}		
if (SignalGroupType[grp] == SignalGroupTypeHOA) {		
numHOATransportChannels += bsNumberOfSignals[grp] + 1;		
}		
}		
}		

5.2.2.3 Core decoder configuration

In 5.2.2.3 replace Table 23 with:

Table 23 — Syntax of mpegh3daExtElementConfig()

Syntax	No. of bits	Mnemonic
mpegh3daExtElementConfig()		
{		
usacExtElementType = escapedValue(4, 8, 16);		
usacExtElementConfigLength = escapedValue(4, 8, 16);		
if (**usacExtElementDefaultLengthPresent**) {	**1**	**uimsbf**
usacExtElementDefaultLength = escapedValue(8, 16, 0) + 1;		
} else {		
usacExtElementDefaultLength = 0;		
}		
usacExtElementPayloadFrag;	**1**	**uimsbf**
switch (usacExtElementType) {		
case ID_EXT_ELE_FILL:		
/* No configuration element */		
break;		
case ID_EXT_ELE_MPEGS:		
SpatialSpecificConfig();		
break;		
case ID_EXT_ELE_SAOC:		
SAOCSpecificConfig();		
break;		
case ID_EXT_ELE_AUDIOPREROLL:		
/* No configuration element */		

[a] The default entry for the usacExtElementType is used for unknown extElementTypes so that legacy decoders can cope with future extensions.

Table 23 *(continued)*

Syntax	No. of bits	Mnemonic
break;		
case ID_EXT_ELE_UNI_DRC:		
mpegh3daUniDrcConfig();		
break;		
case ID_EXT_ELE_OBJ_METADATA:		
ObjectMetadataConfig();		
break;		
case ID_EXT_ELE_SAOC_3D:		
SAOC3DSpecificConfig();		
break;		
case ID_EXT_ELE_HOA:		
HOAConfig();		
break;		
case ID_EXT_ELE_FMT_CNVRTR		
/* No configuration element */		
break;		
case ID_EXT_ELE_MCT:		
MCTConfig();		
break;		
case ID_EXT_ELE_TCC:		
TccConfig();		
break;		
case ID_EXT_ELE_HOA_ENH_LAYER:		
HOAEnhConfig();		
break;		
case ID_EXT_ELE_HREP:		
HREPConfig(current_signal_group);		
break;		
case ID_EXT_ELE_ENHANCED_OBJ_METADATA:		
EnhancedObjectMetadataConfig();		
break;		
case ID_EXT_ELE_PROD_METADATA:		
prodMetadataConfig();		
break;		
default:	[a]	
while (usacExtElementConfigLength--) {		
tmp;	**8**	**uimsbf**
}		
break;		
}		
}		

[a] The default entry for the usacExtElementType is used for unknown extElementTypes so that legacy decoders can cope with future extensions.

5.3.4 Core decoder configuration data elements

In 5.3.4 replace Table 75 with:

Table 75 — Value of usacExtElementType

usacExtElementType	Value
ID_EXT_ELE_FILL	0
ID_EXT_ELE_MPEGS	1
ID_EXT_ELE_SAOC	2
ID_EXT_ELE_AUDIOPREROLL	3
ID_EXT_ELE_UNI_DRC	4
ID_EXT_ELE_OBJ_METADATA	5
ID_EXT_ELE_SAOC_3D	6
ID_EXT_ELE_HOA	7
ID_EXT_ELE_FMT_CNVRTR	8
ID_EXT_ELE_MCT	9
ID_EXT_ELE_TCC	10
ID_EXT_ELE_HOA_ENH_LAYER	11
ID_EXT_ELE_HREP	12
ID_EXT_ELE_ENHANCED_OBJ_METADATA	13
ID_EXT_ELE_PROD_METADATA	14
/* reserved for ISO use */	15-127
/* reserved for use outside of ISO scope */	128 and higher
NOTE Application-specific usacExtElementType values are mandated to be in the space reserved for use outside of ISO scope. These are skipped by a decoder as a minimum of structure is required by the decoder to skip these extensions.	

In 5.3.4 replace Table 76 with:

Table 76 — Interpretation of data blocks for extension payload decoding

usacExtElementType	The concatenated usacExtElementSegmentData represents:
ID_EXT_ELE_FILL	Series of **fill_byte**
ID_EXT_ELE_MPEGS	SpatialFrame() as defined in ISO/IEC 23003-1
ID_EXT_ELE_SAOC	SAOCFrame() as defined in ISO/IEC 23003-2
ID_EXT_ELE_AUDIOPREROLL	AudioPreRoll()
ID_EXT_ELE_UNI_DRC	uniDrcGain() as defined in ISO/IEC 23003-4
ID_EXT_ELE_OBJ_METADATA	objectMetadataFrame()
ID_EXT_ELE_SAOC_3D	Saoc3DFrame()
ID_EXT_ELE_HOA	HOAFrame()
ID_EXT_ELE_FMT_CNVRTR	FormatConverterFrame()
ID_EXT_ELE_MCT	MultichannelCodingFrame()
ID_EXT_ELE_TCC	TccGroupOfSegments()
ID_EXT_ELE_HOA_ENH_LAYER	HOAEnhFrame()
ID_EXT_ELE_HREP	HREPFrame(outputFrameLength, current_signal_group)
ID_EXT_ELE_ENHANCED_OBJ_METADATA	EnhancedObjectMetadataFrame()

Table 76 *(continued)*

usacExtElementType	The concatenated usacExtElementSegmentData represents:
ID_EXT_ELE_PROD_METADATA	prodMetadataFrame()
unknown	Unknown data. The data block shall be discarded.

12.2.1 Configuration of HOA elements

In subclause 12.2.1 replace Table 188 with:

Table 188 — Syntax of HOADecoderConfig()

Syntax	No. of bits	Mnemonic
HOADecoderConfig(numHOATransportChannels)		
{		
MinAmbHoaOrder = escapedValue(3,5,0) – 1;	**3,8**	**uimsbf**
MinNumOfCoeffsForAmbHOA = (MinAmbHoaOrder + 1)^2;		
NumOfAdditionalCoders = numHOATransportChannels –		
MinNumOfCoeffsForAmbHOA;		
NumLayers = 1;		
NumHOAChannelsLayer[0] = numHOATransportChannels;		
if(**SingleLayer** == 0){	**1**	**bslbf**
HOALayerChBits = ceil(log2(NumOfAdditionalCoders));		
NumHOAChannelsLayer[0] = **codedLayerCh** +	**HOALayerChBits**	**uimsbf**
MinNumOfCoeffsForAmbHOA;		
remainingCh = numHOATransportChannels –		
NumHOAChannelsLayer[0];		
while (remainingCh>1) {		
HOALayerChBits = ceil(log2(remainingCh));		
NumHOAChannelsLayer[NumLayers] =	**HOALayerChBits**	**uimsbf**
NumHOAChannelsLayer[NumLayers-1] +		
codedLayerCh + 1;		
remainingCh = numHOATransportChannels –		
NumHOAChannelsLayer[NumLayers];		
NumLayers++;		
}		
if (remainingCh) {		
NumHOAChannelsLayer[NumLayers] =		
numHOATransportChannels;		
NumLayers++;		
}		
}		
CodedSpatialInterpolationTime;	**3**	**uimsbf**
SpatialInterpolationMethod;	**1**	**bslbf**
NOTE MinAmbHoaOrder = 30 ... 37 are reserved. HOAFrameLengthIndicator = 3 is reserved. CodedVVecLength = 3 is reserved.		

Table 188 *(continued)*

Syntax	No. of bits	Mnemonic
CodedVVecLength;	**2**	**uimsbf**
MaxGainCorrAmpExp;	**3**	**uimsbf**
HOAFrameLengthIndicator;	**2**	**uimsbf**
if(MinAmbHoaOrder < HoaOrder) {		
DiffOrderBits = ceil(log2(HoaOrder- MinAmbHoaOrder+1))		
MaxHoaOrderToBeTransmitted = **DiffOrder** +	**DiffOrderBits**	**uimsbf**
MinAmbHoaOrder;		
}		
else {		
MaxHoaOrderToBeTransmitted = HoaOrder;		
}		
MaxNumOfCoeffsToBeTransmitted =		
(MaxHoaOrderToBeTransmitted + 1)^2;		
MaxNumAddActiveAmbCoeffs =		
MaxNumOfCoeffsToBeTransmitted		
- MinNumOfCoeffsForAmbHOA;		
VqConfBits = ceil (log2(ceil(log2(NumOfHoaCoeffs+1))));		
NumVVecVqElementsBits;	**VqConfBits**	**uimsbf**
if(MinAmbHoaOrder == 1) {		
UsePhaseShiftDecorr;	**1**	**bslbf**
}		
if(SingleLayer==1) {		
HOADecoderEnhConfig();		
}		
AmbAsignmBits = ceil(log2(MaxNumAddActiveAmbCoeffs));		
ActivePredIdsBits = ceil(log2(NumOfHoaCoeffs));		
i = 1;		
while(i * ActivePredIdsBits		
+ ceil(log2(i)) < NumOfHoaCoeffs){		
i++;		
}		
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;		
}		
}		
NOTE MinAmbHoaOrder = 30 … 37 are reserved. HOAFrameLengthIndicator = 3 is reserved. CodedVVecLength = 3 is reserved.		

14.2.1 Main MHAS syntax elements

In 14.2.1 replace Table 220 with:

Table 220 — Syntax of MHASPacketPayload()

Syntax	No. of bits	Mnemonic
MHASPacketPayload(MHASPacketType)		
{		
switch (MHASPacketType) {		
case PACTYP_SYNC:		
0xA5; /* syncword*/	**8**	**uimsbf**
break;		
case PACTYP_MPEGH3DACFG:		
mpegh3daConfig();		
break;		
case PACTYP_MPEGH3DAFRAME:		
mpegh3daFrame();		
break;		
case PACTYP_AUDIOSCENEINFO:		
mae_AudioSceneInfo();		
break;		
case PACTYP_FILLDATA:		
for (i=0; i< MHASPacketLength; i++) {		
mhas_fill_data_byte(i);	**8**	**bslbf**
}		
break;		
case PACTYP_SYNCGAP:		
syncSpacingLength = escapedValue(16,24,24);	**16,40,64**	**uimsbf**
break;		
case PACTYP_MARKER:		
for (i=0; i< MHASPacketLength; i++) {		
marker_byte(i);	**8**	**bslbf**
}		
break;		
case PACTYP_CRC16:		
mhasParity16Data;	**16**	**bslbf**
break;		
case PACTYP_CRC32:		
mhasParity32Data;	**32**	**bslbf**
break;		
case PACTYP_GLOBAL_CRC16:		
global_CRC_type;	**2**	**bslbf**
numProtectedPackets;	**6**	**bslbf**
mhasParity16Data;	**16**	**bslbf**
break;		

Table 220 *(continued)*

Syntax	No. of bits	Mnemonic
case PACTYP_ GLOBAL_CRC32:		
global_CRC_type;	**2**	**bslbf**
numProtectedPackets;	**6**	**bslbf**
mhasParity32Data;	**32**	**bslbf**
break;		
case PACTYP_DESCRIPTOR:		
for (i=0; i< MHASPacketLength; i++) {		
mhas_descriptor_data_byte(i);	**8**	**bslbf**
}		
break;		
case PACTYP_USERINTERACTION:		
mpegh3daElementInteraction();		
break;		
case PACTYP_LOUDNESS_DRC:		
mpegh3daLoudnessDrcInterface();		
break;		
case PACTYP_BUFFERINFO:		
mhas_buffer_fullness_present	**1**	**uimsbf**
if (mhas_buffer_fullness_present)		
mhas_buffer_fullness = escapedValue(15,24,32);	**15,39,71**	**uimsbf**
}		
break;		
case PACTYP_AUDIOTRUNCATION:		
audioTruncationInfo();		
break;		
case PACTYP_GENDATA:		
GenDataPayload();		
break;		
case PACTYP_EARCON:		
earconInfo();		
break;		
case PACTYP_PCMCONFIG:		
pcmDataConfig();		
break;		
case PACTYP_PCMDATA:		
pcmDataPayload();		
break;		
case PACTYP_LOUDNESS:		
mpegh3daLoudnessInfoSet();		
break;		
}		
ByteAlign();		
}		

14.3.1 mpeghAudioStreamPacket()

In 14.3.1 replace Table 223 with:

Table 223 — Value of MHASPacketType

MHASPacketType	Value
PACTYP_FILLDATA	0
PACTYP_MPEGH3DACFG	1
PACTYP_MPEGH3DAFRAME	2
PACTYP_AUDIOSCENEINFO	3
/ reserved for ISO use */*	4-5
PACTYP_SYNC	6
PACTYP_SYNCGAP	7
PACTYP_MARKER	8
PACTYP_CRC16	9
PACTYP_CRC32	10
PACTYP_DESCRIPTOR	11
PACTYP_USERINTERACTION	12
PACTYP_LOUDNESS_DRC	13
PACTYP_BUFFERINFO	14
PACTYP_GLOBAL_CRC16	15
PACTYP_GLOBAL_CRC32	16
PACTYP_AUDIOTRUNCATION	17
PACTYP_GENDATA	18
PACTYP_EARCON	19
PACTYP_PCMCONFIG	20
PACTYP_PCMDATA	21
PACTYP_LOUDNESS	22
/ reserved for ISO use */*	23-127
/ reserved for use outside of ISO scope */*	128-261
/ reserved for ISO use */*	262-389
/ reserved for use outside of ISO scope */*	390-517
NOTE Application-specific MHASPacketType values are mandated to be in the space reserved for use outside of ISO scope. These are skipped by a decoder as a minimum of structure is required by the decoder to skip these extensions.	

14.3.2 MHASPacketPayload()

At the end of subclause 14.3.2 add:

earconInfo()	Earcon Info structure as defined in 28.2.
pcmDataConfig()	PCM data configuration structure as defined in 28.2.
pcmDataPayload()	PCM data payload structure as defined in 28.2.
mpegh3daLoudnessInfoSet()	Loudness metadata structure as defined in 6.3.1.