**ISO/IEC JTC 1/SC 29**

Date 2017-10-27

**ISO 23008-3:2015/PDAM 5:201x (X)**

ISO/IEC JTC 1/SC 29/WG 11

Secretariat: JISC

**Information technology — High efficiency coding and media delivery in heterogeneous environments — Part 3: 3D audio, AMENDMENT 1: Audio Metadata Enhancements**

PDAM

**Warning for WDs and CDs**

This document is not an ISO International Standard. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an International Standard.

Recipients of this draft are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

© ISO 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

[Foreword iv](#_Toc485815077)

[Introduction iv](#_Toc485815078)

[1 Scope *(mandatory)* 4](#_Toc485815079)

[2 Normative references *(mandatory)* 4](#_Toc485815080)

[3 Terms and definitions *(mandatory)* 4](#_Toc485815081)

[4 Clause title autonumber 4](#_Toc485815082)

[5 Clause title 4](#_Toc485815083)

[5.1 Subclause autonumber 4](#_Toc485815084)

[5.1.1 Subclause autonumber 4](#_Toc485815085)

[6 Clause title 4](#_Toc485815086)

[Annex A (informative) Annex title e.g. Example of a figure and a table 4](#_Toc485815087)

[Bibliography 4](#_Toc485815088)

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

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](https://www.iso.org/directives-and-policies.html)).

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](https://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 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 the following URL: [www.iso.org/iso/foreword.html](https://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.*

**Information technology — High efficiency coding and media delivery in heterogeneous environments — Part 3: 3D audio, AMENDMENT 1: Metadata**

1. **Changes to the text of ISO/IEC 23008-3:2015**

*Some modifications are highlighted by coloured* *background as follows. These highlights shall be removed upon integration into ISO/IEC 23008-3.*

*YELLOW Cross-references, subclause and table value numbers that need to be checked and aligned with ISO/IEC 23008-3:2015.*

*GRAY Changes relative to ISO/IEC 23008-3:2015 that are highlighted for better visibility.*

## Production metadata

*Replace Table 23 by:*

**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 \*/ |  |  |
| 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: | **NOTE** |  |
| while (usacExtElementConfigLength--) { |  |  |
| **tmp**; | **8** | **uimsbf** |
| } |  |  |
| break; |  |  |
| } |  |  |
| } |  |  |
| NOTE: The default entry for the usacExtElementType is used for unknown extElementTypes so that legacy decoders can cope with future extensions. | | |

*Replace Table 75 by:*

**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. | |

*Replace Table 76 by:*

**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() |
| ID\_EXT\_ELE\_PROD\_METADATA | prodMetadataFrame() |
| unknown | unknown data. The data block shall be discarded. |

*Add new clause after clause 26:*

1. **Production Metadata Decoding**
   1. **Introduction**

Audio metadata originates from production tools and production formats. Audio metadata should be made available in the bit stream to enable a renderer to perform advanced rendering of immersive audio. This clause describes the production metadata and the decoding process thereof.

**27.1.1 Object distance coding**

The object distance is signalled as a 7-bit value allowing coding of values from 0 up to 100m when using an exponential mapping. The resolution of the distance is highest for near positions (<1mm) and lowest in the far positions (around 8.75m). The very low distances below 10mm are considered less important, thus an offset is applied to start the distance coding from 10mm for the second quantized value (=1). The lowest value signals distance = 0.

**27.2.1 Direct headphone signalling**

The **directHeadphone** flag defines that the corresponding signal group of type channels goes to the headphone output directly. The channel group can be mono or stereo, i.e. the **directHeadphone** flag shall be 0 for all signal groups of type channels, which have a different layout than mono or stereo assigned to them. For stereo, the two signals are mixed to left and right headphone channel, directly. For mono, the signal is mixed to left and right headphone channel with a gain factor of 0.707. Over loudspeakers, the signals would come out at the speakers indicated in the CICP Layout index.

The signals flow for the **directHeadphone** channels is modified only if a binaural output signal is generated. For decoding and rendering for loudspeaker playback, no change is needed and the signal is mixed to the output channels according to the rule set of the Format Converter.

When using the channel output interface the **directHeadphone** signal provided to the output interface.  
In case of binaural rendering, channel group signals are processed by DRC1 and then bypassing the Format Converter. The sampling rate of the **directHeadphone** channels is converted to match the output sampling rate. The signal is not mixed and also bypassing the binaural renderer. The signal is delay-aligned to match the delay introduced to the non-directHeadphone signals by the Format converter and the binaural renderer. The signal is mixed to the input of DRC2 after the binaural renderer.

* 1. **Syntax**
     1. **Production Metadata Configuration**

**Table AMD5.1 — Syntax of prodMetadataConfig()**

|  |  |  |
| --- | --- | --- |
| Syntax | No. of bits | Mnemonic |
| prodMetadataConfig() | | |
| { |  |  |
|  |  |  |
| /\* high resolution object distance \*/ |  |  |
| for (gp = 0; gp < numObjectGroups; gp++ ) { /\* NOTE 1 \*/ |  |  |
| **has\_object\_distance**[gp]**;** | **1** | **bslbf** |
| } |  |  |
|  |  |  |
| /\* direct to headphone \*/ |  |  |
| for (gp = 0; gp < numChannelGroups; gp++ ) { /\* NOTE 2 \*/ |  |  |
| **directHeadphone**[gp]; | **1** | **bslbf** |
| } |  |  |
|  |  |  |
| } |  |  |
| NOTE 1: numObjectGroups represents the number of signal groups with signalGroupType == SignalGroupTypeObject as given by the Signals3d() structure in Table 14.  NOTE 2: numChannelGroups represents the number of signal groups with signalGroupType == SignalGroupTypeChannel as given by the Signals3d() structure in Table 14. | | |

* + 1. **Production Metadata Frame**

**Table AMD5.2 — Syntax of prodMetadataFrame()**

|  |  |  |
| --- | --- | --- |
| Syntax | No. of bits | Mnemonic |
| prodMetadataFrame () |  |  |
| { |  |  |
| for ( gp = 0; gp < numObjectGroups; gp++ ) { /\* NOTE 1 \*/ |  |  |
| if ( has\_object\_distance [gp] ) { |  |  |
| **has\_intracoded\_data;** | **1** | **bslbf** |
| if (has\_intracoded\_data) { |  |  |
| intracodedProdMetadataFrame(); |  |  |
| } |  |  |
| else { |  |  |
| dynamicProdMetadataFrame(); |  |  |
| } |  |  |
| } |  |  |
| } |  |  |
| } |  |  |
| NOTE 1: numObjectGroups represents the number of signal groups with signalGroupType == SignalGroupTypeObject as given by the Signals3d() structure in Table 14. | | |

**Table AMD5.3 — Syntax of intracodedProdMetadataFrame();**

|  |  |  |
| --- | --- | --- |
| Syntax | No. of bits | Mnemonic |
| intracodedProdMetadataFrame (); | | |
| { |  |  |
| if (num\_objects>1) { /\* NOTE 1 \*/ |  |  |
| **fixed\_distance;** | **1** | **bslbf** |
| if (fixed\_distance) { |  |  |
| **default\_distance;** | **7** | **tcimsbf** |
| } |  |  |
| else { |  |  |
| **common\_distance;** | **1** | **bslbf** |
| if (common\_distance) { |  |  |
| **default\_distance;** | **7** | **tcimsbf** |
| } |  |  |
| else { |  |  |
| for (o = 0; o < num\_objects; o++) { |  |  |
| **position\_distance**[o]; | **7** | **tcimsbf** |
| } |  |  |
| } |  |  |
| } |  |  |
| } |  |  |
| else { |  |  |
| **position\_distance**[0]**;** | **7** | **tcimsbf** |
| } |  |  |
| } |  |  |
| NOTE 1: num\_objects is equal to the number of objects in the associated signal group. | | |

**Table AMD5.4 — Syntax of dynamicProdMetadataFrame()**

|  |  |  |
| --- | --- | --- |
| Syntax | No. of bits | Mnemonic |
| dynamicProdMetadataFrame () { |  |  |
| **flag\_dist\_absolute;** | **1** | **bslbf** |
| for (o = 0; o < num\_objects; o++) { /\* NOTE 1 \*/ |  |  |
| if (has\_object\_metadata) { /\* NOTE 2 \*/ |  |  |
| singleDynamicProdMetadataFrame(flag\_dist\_absolute); | | |
| } |  |  |
| } |  |  |
| } |  |  |
| NOTE 1: num\_objects is equal to the number of objects in the associated signal group.  NOTE 2: has\_object\_metadata is given by the dynamic\_object\_metadata() structure in Table 142. | | |

**Table AMD5.5 — Syntax of singleDynamicProdMetadataFrame()**

|  |  |  |
| --- | --- | --- |
| Syntax | No. of bits | Mnemonic |
| singleDynamicProdMetadataFrame(flag\_dist\_absolute) { |  |  |
| if (flag\_dist\_absolute) { |  |  |
| if (!fixed\_distance) { /\* NOTE 1 \*/ |  |  |
| **position\_distance**; | **7** | **tcimsbf** |
| } |  |  |
| else { |  |  |
| if (!fixed\_distance) { /\* NOTE 1 \*/ |  |  |
| **flag\_distance;** | **1** | **bslbf** |
| if (flag\_distance) { |  |  |
| **nBitsDistance;** | **3** | **uimsbf** |
| num\_bits = nBitsDistance + 2; |  |  |
| **position\_distance\_difference;** | **min(num\_bits,2)** | **tcimsbf** |
| } |  |  |
| } |  |  |
| } |  |  |
| } |  |  |
| NOTE 1: fixed\_distance given in the preceding intracodedProdMetadataFrame() | | |

* 1. **Semantics**
     1. **Production Metadata Configuration**

**has\_object\_distance** This flag defines if the object distance parameter is signalled in prodMetadataFrame().

**directHeadphone** This flag defines that the corresponding signal group of type channels goes to the headphone output, directly, if the binaural output is rendered. The signals are routed to left and right headphone channel. For mono, the signal is mixed to left and right headphone channel with a gain factor of 0.707.

**has\_intracoded\_data** Flag indicating that the current frame holds intracoded data.

**position\_distance** This field describes the distance between the centre of the head of the listener at the sweet spot position and an object. The field can take values between 0 and 127, which maps to distance values between 10 and 100000 millimetres.

For **position\_distance = 0**, the distance is 0, and

for **position\_distance = 127**, the distance is 100m or above, and

for all other values, the following decoding applies:

distance = distanceOffset + 10 ^ (0,03968223 \* **position\_distance**) -1;

The distanceOffset is 10.

**fixed\_distance** Flag indicating whether the distance value is fixed for all objects.

**common\_distance** Indicates whether a common distance value is used for all objects.

**default\_distance** Defines the value of the common distance for all object.

**flag\_dist\_absolute** Flag indicating whether the values of the components are transmitted differentially or in absolute values.

**flag\_distance** flag per object indicating whether the distance value changes for this intra-frame period.

**nBitsDistance** Defines how many bits are required to represent the differential value minus 2.

**position\_distance\_difference** value of the difference between the linearly interpolated and the actual value of distance.

* 1. **Decoding Process**

The prodMetadataConfig() and the prodMetadataFrame() structures are located in an mpegh3daExtElementConfig and an mpegh3daExtElement structure as defined in Table 23 and 76.

*In subclause 17.10.3.1, extend paragraphs by:*

If an object output interface is provided by an implementation, the following metadata shall be provided via the application specific interface to be evaluated by possible external renderers:

* number of output objects
* information about audio truncation and number of valid PCM frames for the current frame
* OAM metadata
  + dynamic object priority (if available)
  + object position (azimuth, elevation, radius)
  + spread
  + object gain
* Signal group related metadata
  + static group priority
  + “fixed position” flag
* Enhanced object metadata
  + diffuseness
  + divergence and divergence azimuth range
  + exclusion sector metadata
  + Production Metadata

*Replace Table 265 by:*

**Table 265 — Syntax of mpegh3da\_getObjectAudioAndMetadata()**

|  |  |  |
| --- | --- | --- |
| Syntax | No. of bits | Mnemonic |
| mpegh3da\_getObjectAudioAndMetadata() |  |  |
| { |  |  |
| /\* FRAME CONFIGURATION \*/ |  |  |
| **goa\_frameLength;** | **6** | **uimsbf** |
| **goa\_audioTruncation;** | **2** | **bslbf** |
| if (goa\_audioTruncation>0) { |  |  |
| **goa\_numSamples;** | **13** | **uimsbf** |
| } else { |  |  |
| goa\_numSamples = goa\_frameLength << 6; |  |  |
| } |  |  |
|  |  |  |
| /\* OBJECT METADATA \*/ |  |  |
| **goa\_numberOfOutputObjects;** | **9** | **uimsbf** |
| for ( o = 0; o < goa\_numberOfOutputObjects; o++ ) { |  |  |
| **goa\_elementID**[o]; | **9** | **uimsbf** |
| **goa\_hasDynamicObjectPriority**[o]; | **1** | **bslbf** |
| **goa\_hasUniformSpread**[o]; | **1** | **bslbf** |
|  |  |  |
| /\* OAM Data \*/ |  |  |
| **goa\_numOAMframes**[o] | **6** | **uimsbf** |
| for (nf = 0; nf < goa\_numOAMframes[o]; nf++) { |  |  |
| **goa\_objectMetadataPresent**; | **1** | **bslbf** |
| if (goa\_objectMetadataPresent==1){ |  |  |
| **goa\_positionAzimuth**[o][nf]; | **8** | **uimsbf** |
| **goa\_positionElevation**[o][nf]; | **6** | **uimsbf** |
| **goa\_positionRadius**[o][nf]; | **4** | **uimsbf** |
| **goa\_objectGainFactor**[o][nf]; | **7** | **uimsbf** |
|  |  |  |
| if (goa\_hasDynamicObjectPriority[o]) { |  |  |
| **goa\_dynamicObjectPriority**[o][nf]; | **3** | **uimsbf** |
| } |  |  |
|  |  |  |
| if ( goa\_hasUniformSpread[o] ) { |  |  |
| **goa\_uniformSpread**[o][nf]; | **7** | **uimsbf** |
| } else { |  |  |
| **goa\_spreadWidth**[o][nf]; | **7** | **uimsbf** |
| **goa\_spreadHeight**[o][nf]; | **5** | **uimsbf** |
| **goa\_spreadDepth**[o][nf]; | **4** | **uimsbf** |
| } |  |  |
| } |  |  |
| } |  |  |
|  |  |  |
| /\* Signal group related data \*/ |  |  |
| **goa\_fixedPosition**[o]; | **1** | **bslbf** |
| **goa\_groupPriority**[o]; | **3** | **uimsbf** |
|  |  |  |
| /\* Enhanced Object Metadata \*/ |  |  |
| **goa\_diffuseness**[o]; | **7** | **uimsbf** |
| **goa\_divergence**[o]; | **7** | **uimsbf** |
| **goa\_divergenceAzimuthRange**[o]; | **6** | **uimsbf** |
| **goa\_numExclusionSectors**[o]; | **4** | **uimsbf** |
| for ( s = 0; s < goa\_numExclusionSectors[o]; s++) { |  |  |
| **goa\_usePredefinedSector**[o][s]; | **1** | **bslbf** |
| if ( goa\_usePredefinedSector[o][s] ) { |  |  |
| **goa\_excludeSectorIndex**[o][s]; | **4** | **uimsbf** |
| } else { |  |  |
| **goa\_excludeSectorMinAzimuth**[o][s]; | **7** | **uimbsf** |
| **goa\_excludeSectorMaxAzimuth**[o][s] | **7** | **uimbsf** |
| **goa\_excludeSectorMinElevation**[o][s]; | **5** | **uimbsf** |
| **goa\_excludeSectorMaxElevation**[o][s] | **5** | **uimbsf** |
| } |  |  |
| } */\* for ( s = 0; s < goa\_numExclusionSectors[o]; s++) \*/* |  |  |
| } */\* for ( o = 0; o < goa\_numberOfOutputObjects; o++ ) \*/* |  |  |
|  |  |  |
|  |  |  |
| /\* GOA EXTENSION ELEMENTS \*/ |  |  |
| **goa\_numberOfExtensionElements;** | **3** | **uimsbf** |
|  |  |  |
| if (goa\_numberOfExtensionElements) |  |  |
| { |  |  |
| for ( ext = 0; ext < goa\_numberOfExtensionElements; ext++ ) { |  |  |
| **goa\_extElementType;** | **3** | **uimbsf** |
| **goa\_extElementLength;** | **10** | **uimsbf** |
|  |  |  |
| switch (goa\_extElementType) { |  |  |
| case ID\_EXT\_GOA\_PROD\_METADATA: |  |  |
| goa\_Production\_Metadata(); |  |  |
| break; |  |  |
| default: |  |  |
| break; |  |  |
| } |  |  |
| } |  |  |
| } |  |  |
| } |  |  |

*Add new tables after Table 265:*

**Table AMD5.6 — Syntax of goa\_Production\_Metadata ()**

|  |  |  |
| --- | --- | --- |
| Syntax | No. of bits | Mnemonic |
| goa\_Production\_Metadata() |  |  |
| { |  |  |
| /\* PRODUCTION METADATA CONFIGURATION \*/ |  |  |
| **goa\_** **hasObjectDistance;** | **1** | **bslbf** |
| if (goa\_hasObjectDistance) { |  |  |
| for ( o = 0; o < goa\_numberOfOutputObjects; o++ ) { |  |  |
| **goa\_bsObjectDistance**[o] | **7** | **uimsbf** |
| } |  |  |
| } |  |  |
| } |  |  |

**Table AMD5.7 — Syntax of goa\_extElementType**

|  |  |
| --- | --- |
| **goa\_extElementType** | **Value** |
| ID\_EXT\_GOA\_PROD\_METADATA | 0 |
| /\* reserved \*/ | 1-7 |

*Add to section* ***17.10.3.3***

**17.10.3.3. Semantics of the interface for object-based metadata (informative)**

**goa\_numberOfExtensionElements** Defines the number of extension elements to the GOA output interface.

**goa\_extElementType** Defines the type of the extension element.

**goa\_extElementLength** Defines the length of the extension element.

**goa\_hasObjectDistance** This flag defines if the object distance parameter is signalled in the production metadata frame.

**goa\_bsObjectDistance** This field describes the distance of an object. The field can take values between 0 and 127, which maps to distance values between 10 and 100000 millimetres.

For **goa\_bsObjectDistance = 0**, the distance is 0, and

for **goa\_bsObjectDistance = 127**, the distance is 100m or above, and

for all other values, the following decoding applies:

distance = distanceOffset + 10 ^ (0,03968223 \* **goa\_bsObjectDistance**) -1;

The distanceOffset is 10.

*Replace in Section 17.10.4.2 Table 267 by:*

**Table 267 — Syntax of mpegh3da\_getChannelMetadata()**

|  |  |  |
| --- | --- | --- |
| Syntax | No. of bits | Mnemonic |
| mpegh3da\_getChannelMetadata() |  |  |
| { |  |  |
| /\* FRAME CONFIGURATION \*/ |  |  |
| **gca\_frameLength;** | **6** | **uimsbf** |
| **gca\_audioTruncation;** | **2** | **bslbf** |
| if (gca\_audioTruncation>0) { |  |  |
| **gca\_numSamples;** | **13** | **uimsbf** |
| } else { |  |  |
| gca\_numSamples = gca\_frameLength << 6; |  |  |
| } |  |  |
|  |  |  |
| /\* CHANNEL METADATA \*/ |  |  |
| **gca\_numberOfOutputChannelGroups**; | **9** | **uimsbf** |
| for ( cGrp = 0; cGrp < gca\_numberOfOutputChannelGroups; cGrp ++ ) { | |  |
| **gca\_numberOfChannels**[cGrp] | **16** | **uimsbf** |
| gca\_channelLayout[cGrp] = SpeakerConfig3d(); |  |  |
|  |  |  |
| for ( nChn = 0; nChn < gca\_numberOfChannels[cGrp]; nChn++ { | | |
| **gca\_elementID**[cGrp][nChn]; | **9** | **uimsbf** |
| } |  |  |
|  |  |  |
| /\* TRACKING-RELATED METADATA \*/ |  |  |
| **gca\_fixedChannelsPosition**[cGrp]; | **1** | **bslbf** |
|  |  |  |
| /\* GROUP-RELATED METADATA \*/ |  |  |
| **gca\_groupPriority**[cGrp]; | **3** | **uimsbf** |
| **gca\_channelGain**[cGrp]; | **8** | **uimsbf** |
|  |  |  |
| /\* DOWNMIX MATRIX ELEMENT \*/ |  |  |
| **gca\_downmixAvailable** | **1** | **bslbf** |
| if (gca\_downmixAvailable) { |  |  |
| gca\_downmixConfig(); |  |  |
| } |  |  |
|  |  |  |
| /\* PRODUCTION METADATA \*/ |  |  |
| gca\_Production\_Metadata() |  |  |
| } |  |  |
| } |  |  |
|  |  |  |

*Add table below*

**Table AMD5.8 — Syntax of gca\_Production\_Metadata ()**

|  |  |  |
| --- | --- | --- |
| Syntax | No. of bits | Mnemonic |
| gca\_Production\_Metadata() |  |  |
| { |  |  |
| /\* PRODUCTION METADATA CONFIGURATION \*/ |  |  |
|  |  |  |
| for (gp = 0; gp < numChannelGroups; gp++ ) { |  |  |
| **gca\_directHeadphone**[gp] |  |  |
| } |  |  |
|  |  |  |

*Add to*

**17.10.4.3. Semantics of the interface for channel-based metadata (informative)**

**gca\_directHeadphone** This flag defines that the corresponding signal group of type objects goes to the headphone output, directly. The signals are routed to left and right headphone channel. For mono, the signal is mixed to left and right headphone channel with a gain factor of 0.707.

*Replace in Annex C.3.1.* *Pre-Processing of the Object Metadata :*

for (o = 0; o < num\_objects; o++)

radius\_scaled[o][n] = 3.0f \* log2(2.0 \* radius[o][n]);

*by*

for (o = 0; o < num\_objects; o++)

radius\_scaled[o][n] = 3.0f \* log2(2.0 \* radius[o][n]),

where the object radius in the encoder should be set to one, i.e., radius[o][n] = 1. If a distance dependent gain change is desired the object gain value gain[o][n] should be used. Alternatively, the distance gain rendering can be done prior to encoding by changing the gain of the PCM input signal of the object.

# Update of Encoder block diagram description

*In Annex C.5.2 Encoder Block Diagram replace:*

The block diagram of Figure C.5 depicts the structure of the MPEG-H HOA encoder. The HOA input signal matrix 𝑪(𝑘) is analyzed and encoded into the spatial coding parameters and the 𝐼 directional and ambient signals 𝒛𝑖(𝑘 −2). The signals 𝒛𝑖(𝑘 −2) are encoded by the MPEG-H 3D Audio Core encoders. The HOA Frame Creater converts the resulting the HOA spatial coding parameters to the HOA payloads HOAConfig() and HOAFrame().

*with:*

The block diagram of Figure C.5 depicts the structure of the MPEG-H HOA encoder. The HOA input signal matrix 𝑪(𝑘) is analyzed and encoded into the spatial coding parameters and the 𝐼 directional and ambient signals 𝒛𝑖(𝑘 −2). The number *I* of signals 𝒛𝑖 is usually much lower than the number of HOA input coefficients 𝑪(𝑘). The signals 𝒛𝑖(𝑘 −2) are encoded by the MPEG-H 3D Audio Core encoder~~s~~. The HOA Frame Creater converts the resulting ~~the~~ HOA spatial coding parameters to the HOA payloads HOAConfig() and HOAFrame().

In some environments, the HOA spatial encoder may be separated from the MPEG-H 3D Audio Core encoder.In this case, an ***HOA Transport Format*** can consist of the spatial coding parameters and the *I* predominant and ambient signals 𝒛𝑖. This HOA Transport Format can be transmitted from the HOA spatial encoder to the MPEG-H 3D Audio Core encoder. Compared with the input HOA 𝑪(𝑘), the HOA Transport Format usually requires a significantly reduced number of transport channels.