INTERNATIONAL STANDARD

ISO/IEC 23008-3

First edition
2015-10-15

AMENDMENT 4
2016-11-01

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

Part 3: **3D audio**

AMENDMENT 4: Carriage of system data

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

Partie 3: Audio 3D

AMENDEMENT 4: Transport de données système

Reference number
ISO/IEC 23008-3:2015/Amd.4:2016(E)

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

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.

The committee responsible for this document is ISO/IEC JTC 1, *Information technology*, 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 4: Carriage of system data

Clause 18

Add the following:

18.6 Carriage of system metadata for the interaction with system engine

The system data is encapsulated in an MHAS Packet as part of an MHAS audio data stream.

18.6.1 Syntax

For the MHAS Packet with MHASPacketType of **PACTYP_SYSMETA,** SysMetaPayload is defined as shown in Table AMD.1.

If a packet with MHASPacketType of **PACTYP_SYSMETA** is present in the MPEG-H 3D Audio MHAS stream, MHAS Packets of type **PACTYP_MPEGH3DAFRAME** shall also be present in the stream. The total bitrate associated with all packets of type **PACTYP_SYSMETA** shall not be greater than 1 % of the audio bitrate, averaged over a 5 s time window.

For example, if the nominal audio bitrate is 512 kb/s and one **PACTYP_SYSMETA** MHAS Packet of length 256 bytes is transmitted each second, the bitrate associated with these packets is 0.4 % of the audio bitrate.

Table — AMD.1 — Syntax of SysMetaPayload

Syntax	No. of bits	Mnemonic
SysMetaPayload ()		
{		
sysType	**8**	**uimsbf**
if (sysType==0) {		
t35Code;	**16···40**	**uimsbf**
}		
else if (sysType==0xFF) {		
uuid;	**128**	**uimsbf**
shortUuid;	**8**	**uimsbf**
}		
sysData;	**var*8**	**uimsbf**
}		

18.6.2 Semantics

sysType This element indicates corresponding reference of the system data as shown in Table AMD.2. This field takes either a dynamically assigned value (see below under shortUUID) or one of two reserved values.

Table — AMD.2 — Value of sysType

Value	Corresponding References
0	ITU T.35 ID follows
1-254	**sysType** is UUID short form (no additional ID follows). See **uuid** and **shortUuid** field semantics below.
255	Full UUID and its short form follow

t35Code For **sysType** value 0, this field shall be the "country code" field (1 or 2 bytes) followed by the "manufacturer code" field (1 or more bytes) as defined by ITU T.35 [2]. The length is variable, but cannot be less than 2 bytes.

uuid A binary encoding of UUID as defined in IETF RFC 4122 [1].

shortUuid For **sysType** values 1-254, the **sysType** value is the short form value of the UUID corresponding to the previous field, **uuid**. The long form identifier of the system type, **uuid**, is dynamically mapped to this short form identifier, **shortUuid**, and this short form identifier can occur later in the same stream to refer to the associated UUID. This technique saves identifier overhead. The **sysType** value 255 both provides **sysData** associated with a UUID, and establishes a mapping between a **shortUuid** and the full **uuid**, such that the **sysType** field in subsequent packets may be equal to **shortUuid** to indicate that the **sysData** in those packets is associated with this previously mapped **uuid**. The value 0 or 255 shall not be used for **shortUuid**. *Note that the system for which the system metadata applies is responsible for ensuring that the frequency of establishment, and the expiry period (if any), of mappings between short form and long form identifiers are appropriate to the application.*

sysData The variable length system data payload. Its length is the bytes that remain from the MHAS Packet length after the preceding fields in SysMetaPayload().

18.6.3 Processing at the MPEG-H 3D Audio Decoder

MPEG-H 3D Audio decoder should extract the MHAS Packet with PACTYP_SYSMETA and, if **sysType**, **t35Code**, **uuid** and **shortUuid** that occur in the SysMetaPayload() are understood, deliver the complete packet (including the MHAS header) in binary to a system engine known to handle such a **sysType**. If the **sysType**, **t35Code**, **uuid** and **shortUuid** fields in the SysMetaPayload() are not understood, the Audio decoder should discard the Packet. Both the length and value of the **uuid** and **t35Code** fields of interest to the decoder are pre-known to the decoder. For **t35Code,** there is the possibility of a length overrun (e.g. checking for a length of 4 bytes when it is only 3 bytes long). Decoders should ensure the comparison length does not exceed the MHAS Packet length. Any overrun on the comparison length will spill harmlessly into the sysData and the comparison will always fail in such conditions due to the progressive nature of the t35Code encoding. That is, if all the bytes match in **t35Code** then it is a match regardless of the **sysData** contents.

2 Updates to MHAS

Table 139 replace:

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_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_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();		

Syntax	No. of bits	Mnemonic
break;		
case PACTYP_BUFFERINFO:		
mhas_buffer_fullness_present	**1**	**uimsbf**
if (mhas_buffer_fullness_present)		
mhas_buffer_fullness = escapedVal-ue(15,24,32);	**15,39,71**	**uimsbf**
}		
break;		
}		
ByteAlign();		
}		

with:

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:		

Syntax	No. of bits	Mnemonic
mhasParity32Data;	**32**	**bslbf**
break;		
case PACTYP_GLOBAL_CRC16:		
global_CRC_type;	**2**	**bslbf**
numProtectedPackets;	**6**	**bslbf**
mhasParity16Data;	**16**	**bslbf**
break;		
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 = escapedVal-	**15,39,71**	**uimsbf**
ue(15,24,32);		
}		
break;		
case PACTYP_AUDIOTRUNCATION:		
audioTruncationInfo();		
break;		
case PACTYP_SYSMETA:		
SysMetaPayload ();		
break;		
}		
ByteAlign();		
}		