INTERNATIONAL STANDARD

ISO/IEC 23008-3

First edition
2015-10-15

AMENDMENT 2
2016-09-01

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

Part 3:
3D audio

AMENDMENT 2: MPEG-H 3D Audio File Format Support

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

Partie 3: Audio 3D

AMENDEMENT 2: Support de format fichier audio 3D MPEG-H

Reference number
ISO/IEC 23008-3:2015/Amd.2:2016(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. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

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.

Amendment 2 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*.

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

Part 3: **3D audio**

AMENDMENT 2: MPEG-H 3D Audio File Format Support

Page 346

Add the following as a new Clause 20

20 Carriage of MPEG-H 3D Audio in ISO base media file format

20.1 General

This clause specifies the carriage of MPEG-H 3D Audio in the ISO base media file format. 20.2 describes the signalling of random access points for Immediate Play-out Frames (IPF) and independently decodable frames (IF). 20.7 describes the additional signalling of dynamic range control and loudness information that might be present in the encoded bitstream. 20.9 describes the additional signalling of audio scene information data that might be present in the encoded bitstream.

20.2 Random Access and Stream Access

Frames that use AudioPreRoll() following the restrictions in 5.5.6 are considered to be Immediate Play-out Frames (IPF) and shall be signalled by means of the sync sample box according to ISO/IEC 14496-12:2015, 8.6.2.

Independently decodable Frames (IF) as described in 5.7 shall be signalled by means of the roll sample group according to ISO/IEC 14496-12.

20.3 Overview of new box structures

mha1, mha2, mhm1, mhm2			*	*sample entry*
	mhaC			*configuration*
	mhaD			*dynamic range and loudness*
	maeM			*multi-stream*
	maeI			*audio scene information*
		maeG	*	*group definition*
		maeS		*switch group definition*
		maeP		*preset definition*
		maeL		*text label definition*

20.4 MHA decoder configuration record

20.4.1 Definition

This clause specifies the decoder configuration information for MPEG-H 3D Audio (MHA) content.

This record contains a version field. This version of the specification defines version 1 of this record. Incompatible changes to the record will be indicated by a change of version number. Readers must not attempt to decode this record or the streams to which it applies if the version number is unrecognised.

20.4.2 Syntax

```
aligned(8) class MHADecoderConfigurationRecord {
   unsigned int(8)    configurationVersion = 1;
   unsigned int(8)    mpegh3daProfileLevelIndication;
   unsigned int(8)    referenceChannelLayout;
   unsigned int(16)   mpegh3daConfigLength;
   bit(8*mpegh3daConfigLength) mpegh3daConfig;
}
```

20.4.3 Semantics

`configurationVersion`	shall be set to 1 in this version of the specification.
`mpegh3daProfileLevelIndication`	defined in 5.2.2.
`referenceChannelLayout`	ChannelConfiguration value defined in ISO/IEC 23001-8.
`mpegh3daConfigLength`	length in bytes of `mpegh3daConfig`.
`mpegh3daConfig`	the MPEGH-H 3DA configuration defined in this part of ISO/IEC 23008.

20.5 MPEG-H Audio Sample Entry

20.5.1 Definition

Box Types: `'mhaC'`, `'mha1'`, `'mha2'`

Container: Sample Table Box (`'stbl'`)

Mandatory: The mha1 box is mandatory

Quantity: One or more sample entries may be present

The `MHASampleEntry` shall contain a `MHAConfigurationBox`, as defined below. This includes the `MHADecoderConfigurationRecord` as defined in 20.4. If the sample entry type is `'mha1'`, multiple streams shall not be used. If the sample entry name is `'mha2'`, multiple streams may be used.

If an `'mha1'` or `'mha2'` `MHASampleEntry` is present, each sample of the appropriate Track shall contain exactly one mpegh3daFrame as defined in this part of ISO/IEC 23008. An optional `MPEG4BitRateBox` may be present in the `MHASampleEntry` to signal the bit rate information of the MHA stream. Extension descriptors that should be inserted into the Elementary Stream Descriptor, when used in MPEG-4, may also be present. Other boxes may be present in the `MHASampleEntry`. When multiple streams are used, the `MHADecoderConfigurationRecord` for each track shall correspond to the appropriate mpegh3daFrame of that track.

The following optional boxes inherited from `AudioSampleEntry` from ISO/IEC 14496-12/Amd 4:2015 shall not be present

— `DownMixInstructions()`

— `DRCCoefficientsBasic()`

— `DRCInstructionsBasic()`

— `DRCCoefficientsUniDRC()`

— `DRCInstructionsUniDRC()`

20.5.2 Syntax

```
class MHAConfigurationBox() extends Box('mhaC') {
   MHADecoderConfigurationRecord MHAConfig;
}
class MPEG4BitRateBox() extends Box('btrt') {
   unsigned int(32)  bufferSizeDB;
   unsigned int(32)  maxBitrate;
   unsigned int(32)  avgBitrate;
}
class MPEG4ExtensionDescriptorsBox() extends Box('m4ds') {
   Descriptor Descr[0 .. 255];
}
MHASampleEntry() extends AudioSampleEntry('mha1') {
   MHAConfigurationBox config;
   MPEG4BitRateBox();                // optional
   MPEG4ExtensionDescriptorsBox (); // optional
}
MHASampleEntry() extends AudioSampleEntry('mha2') {
   MHAConfigurationBox config;
   MPEG4BitRateBox();                // optional
   MPEG4ExtensionDescriptorsBox (); // optional
}
```

20.5.3 Semantics

`ChannelCount`	inherited from `AudioSampleEntry`, shall be set to 0 (inapplicable) The MPEG-H 3D Audio decoder is capable of rendering a scene to any given speaker setup. The `referenceChannelLayout` carried in the `MHADecoder-ConfigurationRecord` shall be used to signal the preferred reproduction layout for this stream and replaces the `ChannelCount`
`config`	defined in 20.4
`Descr`	is a descriptor which should be placed in in the `ElementaryStreamDescriptor` when this stream is used in an MPEG-4 systems context. This does not include `SLConfigDescriptor` or `DecoderConfigDescriptor`, but includes the other descriptors in order to be placed after the `SLConfigDescriptor`
`bufferSizeDB`	gives the size of the decoding buffer for the elementary stream in bytes
`maxBitrate`	gives the maximum rate in bits/second over any window of 1 second
`minBitrate`	gives the average rate in bits/second over any window of 1 second

20.6 MPEG-H Audio MHAS Sample Entry

20.6.1 Definition

Box Types: '`mhm1`', '`mhm2`'

Container: Sample Table Box ('`stbl`')

Mandatory: No

Quantity: One or more sample entries may be present

Especially in streaming or broadcast environments based on, e.g. MPEG-DASH or MPEG-H MMT, the MPEG-H 3D Audio configuration may change at arbitrary positions of the stream and not necessarily only on fragment boundaries. To enable this use-case the '`mhm1`' and '`mhm2`' `MHASampleEntry` provides an in-band configuration mechanism for MPEG-H 3D Audio files.

If an 'mhm1' or 'mhm2' MHASampleEntry is present, each sample of the appropriate Track shall contain exactly one MHAS Packet with the MHASPacketType PACTYP_MPEGH3DAFRAME as defined in Clause 14.

A sample may contain additional MHAS Packets of other types: if present, an MHAS Packet with MHASPacketType PACTYP_MPEGH3DACFG, PACTYP_AUDIOSCENEINFO or PACTYP_AUDIOTRUNCATION shall directly precede the MHAS Packet of Type PACTYP_MPEGH3DAFRAME.

MHAS Packets with the MHASPacketType PACTYPE_CRC16 and PACTYPE_CRC32 shall not be present in any sample. Other MHAS Packets may be present in a sample.

The first sample of the movie and the first sample of every fragment (when applicable) shall contain a MHAS packet with the type PACTYP_MPEGH3DACFG followed by an MHAS packet with the Type PACTYP_AUDIOSCENEINFO if present.

All samples of the movie that contain an MHAS packet of type PACTYP_MPEGH3DACFG shall be sync samples.

If the movie contains a configuration change, i.e. one of the samples of the movie besides the first sample contains an MHAS packet of type PACTYP_MPEGH3DACFG, all sync samples of the movie shall contain an MHAS packet of type PACTYP_MPEGH3DACFG.

If the sample entry type is 'mhm1', multiple streams shall not be used. If the sample entry name is 'mhm2', multiple streams may be used.

Optional boxes may be present in the MHASampleEntry. Optional boxes for the sample entry type 'mhm1' are handled according to the sample entry type is 'mha1', optional boxes for the sample entry type is 'mhm2' are handled according to the sample entry type is 'mha2'.

In contrast to the sample entry types 'mha1' and 'mha2' the MHAConfigurationBox is optional for the sample entry types 'mhm1' and 'mhm2' and not mandatory.

20.6.2 Syntax

```
MHASampleEntry() extends AudioSampleEntry('mhm1') {
   MHAConfigurationBox config;       // optional
   MPEG4BitRateBox();                  // optional
   MPEG4ExtensionDescriptorsBox();   // optional
}
MHASampleEntry() extends AudioSampleEntry('mhm2') {
   MHAConfigurationBox config;       // optional
   MPEG4BitRateBox();                  // optional
   MPEG4ExtensionDescriptorsBox();   // optional
}
```

20.7 Dynamic Range Control and Loudness

20.7.1 MHA Dynamic Range Control and Loudness

20.7.1.1 Definition

Box Type: 'mhaD'

Container: MHA sample entry ('mha1', 'mha2', 'mhm1', 'mhm2')

Mandatory: No

Quantity: Zero or one

This box specifies the dynamic range control and loudness information that may be contained in the MPEG-H 3D Audio (MHA) track. The provided information represents only a subset of the in-stream configuration according to 6.3.

20.7.1.2 Syntax

```
aligned(8) class MHADynamicRangeControlAndLoudnessBox()
   extends FullBox('mhaD', version = 0, 0) {
   unsigned int(2)    reserved = 0;
   unsigned int(6)    drcInstructionsUniDrcCount;
   unsigned int(2)    reserved = 0;
   unsigned int(6)    loudnessInfoCount;
   unsigned int(2)    reserved = 0;
   unsigned int(6)    loudnessInfoAlbumCount;
   unsigned int(3)    reserved = 0;
   unsigned int(5)    downmixIdCount;

   for (i=0; i < drcInstructionsUniDrcCount; i++) {
      unsigned int(6)    reserved = 0;
      unsigned int(2)    drcInstructionsType;
      if (drcInstructionsType == 2) {
         unsigned int(1)    reserved = 0;
         unsigned int(7)    mae_groupID;
      }
      if (drcInstructionsType == 3) {
         unsigned int(3)    reserved = 0;
         unsigned int(5)    mae_groupPresetID;
      }
      unsigned int(2)    reserved = 0;
      unsigned int(6)    drcSetId;
      unsigned int(1)    reserved = 0;
      unsigned int(7)    downmixId;
      unsigned int(5)    reserved = 0;
      unsigned int(3)    additionalDownmixIdCount;
      for (j=0; j < additionalDownmixIdCount; j++) {
         unsigned int(1)    reserved = 0;
         unsigned int(7)    additionalDownmixId;
      }
      unsigned int(16)   drcSetEffect;
      unsigned int(7)    reserved = 0;
      unsigned int(1)    limiterPeakTargetPresent;
      if (limiterPeakTargetPresent == 1) {
         unsigned int(8)    bsLimiterPeakTarget;
      }
      unsigned int(7)    reserved = 0;
      unsigned int(1)    drcSetTargetLoudnessPresent;
      if (drcSetTargetLoudnessPresent == 1) {
         unsigned int(2)    reserved = 0;
         unsigned int(6)    bsDrcSetTargetLoudnessValueUpper;
         unsigned int(2)    reserved = 0;
         unsigned int(6)    bsDrcSetTargetLoudnessValueLower;
      }
      unsigned int(1)    reserved = 0;
      unsigned int(6)    dependsOnDrcSet;
      if (dependsOnDrcSet == 0) {
         unsigned int(1)    noIndependentUse;
      } else {
         unsigned int(1)    reserved = 0;
      }
   }

   for (i=0; i < loudnessInfoCount; i++) {
      unsigned int(6)    reserved = 0;
      unsigned int(2)    loudnessInfoType;
      if (loudnessInfoType == 1 || loudnessInfoType == 2) {
         unsigned int(1)    reserved = 0;
         unsigned int(7)    mae_groupID;
      } else if (loudnessInfoType == 3) {
         unsigned int(3)    reserved = 0;
         unsigned int(5)    mae_groupPresetID;
      }
      LoudnessBaseBox();
   }

   for (i=0; i < loudnessInfoAlbumCount; i++) {
        LoudnessBaseBox();
    }

    for (i=0; i < downmixIdCount; i++) {
        unsigned int(1)    reserved = 0;
        unsigned int(7)    downmixId;
        unsigned int(2)    downmixType;
        unsigned int(6)    CICPspeakerLayoutIdx;
    }
}
```

20.7.1.3 Semantics

`drcInstructionsUniDrcCount`	number of drcInstructions in the MHA track
`loudnessInfoCount`	number of loudnessInfo blocks in the MHA track
`loudnessInfoAlbumCount`	number of loudnessInfoAlbum blocks in the MHA track
`downmixIdCount`	number of downmixId definitions in the MHA track
`drcInstructionsType`	defined in 6.3 a value of '1' is not defined
`mae_groupID`	defined in 15.3
`mae_groupPresetID`	defined in 15.3
`drcSetId`	defined in ISO/IEC 23003-4:2015, Annex A
`downmixId`	defined in 5.3.5
`additionalDownmixId`	defined in ISO/IEC 23003-4:2015, Annex A
`drcSetEffect`	defined in ISO/IEC 23003-4:2015, Annex A
`bsLimiterPeakTarget`	defined in ISO/IEC 23003-4:2015, Annex A
`bsDrcSetTargetLoudnessValueUpper`	defined in ISO/IEC 23003-4:2015, Annex A
`bsDrcSetTargetLoudnessValueLower`	defined in ISO/IEC 23003-4:2015, Annex A
`dependsOnDrcSet`	defined in ISO/IEC 23003-4:2015, Annex A
`noIndependentUse`	defined in ISO/IEC 23003-4:2015, Annex A
`downmixType`	defined in 5.3.5
`CICPspeakerLayoutIdx`	defined in 5.3.5
`LoudnessBox()`	defined in ISO/IEC 14496-12:2012/Amd.4:2015

20.8 MHA Multi-Stream Signalling

20.8.1 Definition

Box Type: `'maeM'`

Container: MHA sample entry (`'mha1'`, `'mha2'`, `'mhm1'`, `'mhm2'`)

Mandatory: No

Quantity: Zero or one

This box provides information on the location of each mae_groupID in case of splitting the audio scene over multiple streams or files. If multiple streams are used, this box shall be present.

20.8.2 Syntax

```
aligned(8) class MHAMultiStreamBox()
   extends FullBox('maeM', version=0, 0) {
   unsigned int(1) isMainStream;
   unsigned int(7) thisStreamID;

   if (isMainStream) {
      unsigned int(1)    reserved = 0;
      unsigned int(7)    mae_numGroups;
      unsigned int(1)    reserved = 0;
      unsigned int(7)    numAuxiliaryStreams;

      for (i=0; i< mae_numGroups; i++) {
         unsigned int(7)    mae_groupID;
         unsigned int(1)    isInMainStream;
         if (!isInMainStream) {
            unsigned int(1)    reserved = 0;
            unsigned int(7)    auxiliaryStreamID;
         }
      }
   }
}
```

20.8.3 Semantics

`isMainStream`	flag indicating if this is the main stream
`thisStreamID`	unique ID of the audio stream in the scope of all available streams in the MHA scene
`mae_numGroups`	total number of groups in the MHA scene. This value can have a value between 1 and 127, a minimum number of 1 and a maximum number of 127 groups. This number shall be equal to `mae_numGroups` in `MHA–GroupDefinitionBox()`
`numAuxiliaryStreams`	total number of auxiliary streams available
`mae_groupID`	`mae_groupID` (see 15.3) the loop instance refers to
`isInMainStream`	if this flag is set to 1, the audio data related to the group (indicated by `mae_groupID`) is present in the main stream, otherwise the data is transmitted in an auxiliary stream
`auxiliaryStreamID`	in case the audio data identified by `mae_groupID` is an auxiliary stream, this integer identifies the respective auxiliary stream

20.9 Audio Scene Information

20.9.1 MHA Group Definition

20.9.1.1 Definition

Box Type: `'maeG'`

Container: MHA scene information (`'maeI'`)

Mandatory: Yes

Quantity: Zero or one

This box provides information about interactivity and priority of groups contained in an MPEG-H 3D Audio (MHA) track.

20.9.1.2 Syntax

```
aligned(8) class MHAGroupDefinitionBox()
   extends FullBox('maeG', version = 0, 0) {

   unsigned int(8)   mae_audioSceneID;

   unsigned int(1)   reserved = 0;
   unsigned int(7)   mae_numGroups;
   for (i=0; i < mae_numGroups; i++) {
      unsigned int(1)   reserved = 0;
      unsigned int(7)   mae_groupID;
      unsigned int(3)   reserved = 0;
      unsigned int(1)   mae_allowOnOff;
      unsigned int(1)   mae_defaultOnOff;
      unsigned int(1)   mae_allowPositionInteractivity;
      unsigned int(1)   mae_allowGainInteractivity;
      unsigned int(1)   mae_hasContentLanguage;
      unsigned int(4)   reserved = 0;
      unsigned int(4)   mae_contentKind;

      if (mae_allowPositionInteractivity == 1) {
         unsigned int(1)   reserved = 0;
         unsigned int(7)   mae_interactivityMinAzOffset;
         unsigned int(1)   reserved = 0;
         unsigned int(7)   mae_interactivityMaxAzOffset;
         unsigned int(3)   reserved = 0;
         unsigned int(5)   mae_interactivityMinElOffset;
         unsigned int(3)   reserved = 0;
         unsigned int(5)   mae_interactivityMaxElOffset;
         unsigned int(4)   mae_interactivityMinDistFactor;
         unsigned int(4)   mae_interactivityMaxDistFactor;
      }

      if (mae_allowGainInteractivity == 1) {
         unsigned int(2)   reserved = 0;
         unsigned int(6)   mae_interactivityMinGain;
         unsigned int(3)   reserved = 0;
         unsigned int(5)   mae_interactivityMaxGain;
      }

      if (mae_hasContentLanguage == 1) {
         unsigned int(8)   mae_contentLanguage;
      }
   }
}
```

20.9.1.3 Semantics

`mae_audioSceneID`	defined in 15.3
`mae_numGroups`	defined in 15.3
`mae_groupID`	defined in 15.3
`mae_allowOnOff`	defined in 15.3
`mae_defaultOnOff`	defined in 15.3
`mae_allowPositionInteractivity`	defined in 15.3
`mae_allowGainInteractivity`	defined in 15.3
`mae_hasContentLanguage`	defined in 15.3

`mae_contentKind`	defined in 15.3
`mae_interactivityMinAzOffset`	defined in 15.3
`mae_interactivityMaxAzOffset`	defined in 15.3
`mae_interactivityMinElOffset`	defined in 15.3
`mae_interactivityMaxElOffset`	defined in 15.3
`mae_interactivityMinDistFactor`	defined in 15.3
`mae_interactivityMaxDistFactor`	defined in 15.3
`mae_interactivityMinGain`	defined in 15.3
`mae_interactivityMaxGain`	defined in 15.3
`mae_ContentLanguage`	defined in 15.3

20.9.2 MHA Switch Group Definition

20.9.2.1 Definition

Box Type: ‘`maeS`’

Container: MHA scene information (‘`maeI`’)

Mandatory: No

Quantity: Zero or one

20.9.2.2 Syntax

```
aligned(8) class MHASwitchGroupDefinitionBox()
   extends FullBox('maeS', version = 0, 0) {

   unsigned int(3)  reserved = 0;
   unsigned int(5)  mae_numSwitchGroups;
   for (i=0; i < mae_numSwitchGroups; i++) {
      unsigned int(3)      reserved = 0;
      unsigned int(5)      mae_switchGoupID;

      unsigned int(3)      reserved = 0;
      unsigned int(5)      mae_bsSwitchGroupNumMembers;
      for (j=0; j < mae_bsSwitchGroupNumMembers; j++) {
         unsigned int(1)   reserved = 0;
         unsigned int(7)   mae_switchGroupMemberID;
      }

      unsigned int(1)      reserved = 0;
      unsigned int(7)      mae_switchGroupDefaultGroupID;
   }
}
```

20.9.2.3 Semantics

`mae_numSwitchGroups`	defined in 15.3
`mae_switchGroupID`	defined in 15.3
`mae_switchGroupAllowOnOff`	defined in 15.3

`mae_switchGroupDefaultOnOff`	defined in 15.3 If `mae_switchGroupAllowOnOff` is 0, then `mae_switchGroupDefaultOnOff` shall be 0
`mae_bsSwitchGroupNumMembers`	defined in 15.3
`mae_switchGroupMemberID`	defined in 15.3
`mae_switchGroupDefaultGroupID`	defined in 15.3

20.9.3 MHA Group Preset Definition

20.9.3.1 Definition

Box Type: `'maeP'`

Container: MHA scene information (`'maeI'`)

Mandatory: No

Quantity: Zero or one

This box provides information about group presets contained in an MPEG-H 3D Audio (MHA) track. A preset is a collection of groups which cover a common use-case. In addition, a preset can be used to enable advanced DRC for audio object scenes.

20.9.3.2 Syntax

```
aligned(8) class MHAGroupPresetDefinitionBox()
   extends FullBox('maeP', version=0, 0) {

   unsigned int(3)    reserved = 0;
   unsigned int(5)    mae_numGroupPresets;
   for (i=0; i < mae_numGroupPresets; i++) {
      unsigned int(3)    reserved = 0;
      unsigned int(5)    mae_groupPresetID;
      unsigned int(3)    reserved = 0;
      unsigned int(5)    mae_groupPresetKind;

      unsigned int(8)    numGroupPresetConditions;
      for (j=0; j < numGroupPresetConditions; j++) {
         unsigned int(7) mae_groupPresetGroupID;
         unsigned int(1) mae_groupPresetConditionOnOff;
         if (mae_groupPresetConditionOnOff == 1) {
            unsigned int(4)    reserved = 0;
            unsigned int(1)    mae_groupPresetDisableGainInteractivity;
            unsigned int(1)    mae_groupPresetGainFlag;
            unsigned int(1)    mae_groupPresetDisablePositionInteractivity;
            unsigned int(1)    mae_groupPresetPositionFlag;
            if (mae_groupPresetGainFlag == 1) {
               unsigned int(8)    mae_groupPresetGain;
            }
            if (mae_groupPresetPositionFlag == 1) {
               unsigned int(8)    mae_groupPresetAzOffset;
               unsigned int(8)    mae_groupPresetElOffset;
               unsigned int(8)    mae_groupPresetDistFactor;
            }
         }
      }
   }
}
```

20.9.3.3 Semantics

`mae_numGroupPresets`	defined in 15.3
`mae_groupPresetID`	defined in 15.3
`mae_groupPresetKind`	defined in 15.3
`numGroupPresetConditions`	number of group preset conditions
`mae_groupPresetGroupID`	defined in 15.3
`mae_groupPresetConditionOnOff`	defined in 15.3
`mae_groupPresetDisableGainInteractivity`	defined in 15.3
`mae_groupPresetGainFlag`	defined in 15.3
`mae_groupPresetDisablePositionInteractivity`	defined in 15.3
`mae_groupPresetPositionFlag`	defined in 15.3
`mae_groupPresetGain`	defined in 15.3
`mae_groupPresetAzOffset`	defined in 15.3
`mae_groupPresetElOffset`	defined in 15.3
`mae_groupPresetDistFactor`	defined in 15.3

20.9.4 MHA Group Description Text Label

20.9.4.1 Definition

Box Type: `'maeL'`

Container: MHA scene information (`'mael'`)

Mandatory: No

Quantity: Zero or one

20.9.4.2 Syntax

```
aligned(8) class MHAGroupDescrTextLabelBox()
   extends FullBox('maeL', version = 0, 0) {

   unsigned int(4)   reserved = 0;
   unsigned int(4)   numDescLanguage;
   for (j=0; j < numDescLanguage; j++) {
      unsinged int(24)   descriptionLanguage;

      unsigned int(1)   reserved = 0;
      unsigned int(7)   numGroupDescriptions;
      for (i=0; i < numGroupDescriptions; i++) {
         unsigned int(1)   reserved = 0;
         unsigned int(7)   mae_descritptionGroupID;

         unsigned int(8)   groupDescriptionDataLength;
         for (c=0; c < groupDescriptionDataLength; c++) {
            groupDescriptionData;
         }
      }

      unsigned int(3)   reserved = 0;
      unsigned int(5)   numSwitchGroupDescriptions;
```

```
        for (i=0; i < numSwitchGroupDescriptions; i++) {
           unsigned int(3)    reserved = 0;
           unsigned int(5)    mae_descriptionSwitchGroupID;

           unsigned int(8)    switchGroupDescriptionDataLength;
           for (c=0; c < switchGroupDescriptionDataLength; c++) {
              switchGroupDescriptionData;
           }
        }

        unsigned int(3)    reserved = 0;
        unsigned int(5)    numGroupPresets;
        for (i=0; i < numGroupPresets; i++) {
           unsigned int(3)    reserved = 0;
           unsigned int(5)    mae_descriptionGroupPresetID;

           unsigned int(8)    groupPresetDescriptionDataLength;
           for (c=0; c < groupPresetDescriptionDataLength; c++) {
              groupPresetDescriptionData;
           }
        }
     }
}
```

20.9.4.3Semantics

`numDescLanguage`	number of available languages
`descriptionLanguage`	language of the description text

The field contains a 3-character code as specified in ISO 639-2. Both ISO 639-2/B and ISO 639-2/T may be used. Each character is coded into 8 bits according to ISO/IEC 8859-1 and inserted in order into the 24-bit field. EXAMPLE: French has 3-character code “fre”, which is coded as: “0110 0110 0111 0010 0110 0101”

`numGroupDescription`	number of group definition blocks
`mae_descriptionGroupID`	defined in 15.3
`groupDescriptionDataLength`	length in bytes of groupDescriptionData
`groupDescriptionData`	description of a group

A string describing the content by a high-level description. The format shall follow UTF-8 encoding according to ISO/IEC 10646

`numSwitchGroupDescriptions`	number of switch group definition blocks
`mae_descriptionSwitchGroupID`	defined in 15.3
`switchGroupDescriptionDataLength`	length in bytes of switchGroupDescriptionData
`switchGroupDescriptionData`	description of a switch group

A string describing the content by a high-level description. The format shall follow UTF-8 encoding according to ISO/IEC 10646

`numGroupPresets`	number of group preset definition blocks
`mae_descriptionGroupPresetID`	defined in 15.3
`groupPresetDescriptionDataLength`	length in bytes of groupPresetDescriptionData
`groupPresetDescriptionData`	description of a group preset