ETSI TS 126 258 V19.2.0 (2026-04)

TECHNICAL SPECIFICATION

LTE;
5G;
Codec for Immersive Voice and Audio Services (IVAS);
C code (floating-point)
(3GPP TS 26.258 version 19.2.0 Release 19)

Reference
RTS/TSGS-0426258vj20

Keywords
5G,LTE

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - APE 7112B
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° w061004871

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The declarations pertaining to these essential IPRs, if any, are publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI IPR online database.

Pursuant to the ETSI Directives including the ETSI IPR Policy, no investigation regarding the essentiality of IPRs, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

DECT™, **PLUGTESTS™**, **UMTS™** and the ETSI logo are trademarks of ETSI registered for the benefit of its Members. **3GPP™**, **LTE™** and **5G™** logo are trademarks of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners. **oneM2M™** logo is a trademark of ETSI registered for the benefit of its Members and of the oneM2M Partners. **GSM**® and the GSM logo are trademarks registered and owned by the GSM Association.

Legal Notice

This Technical Specification (TS) has been produced by ETSI 3rd Generation Partnership Project (3GPP).

The present document may refer to technical specifications or reports using their 3GPP identities. These shall be interpreted as being references to the corresponding ETSI deliverables.

The cross reference between 3GPP and ETSI identities can be found at 3GPP to ETSI numbering cross-referencing.

Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

Contents

Foreword

This Technical Specification has been produced by the 3rd Generation Partnership Project (3GPP).

The contents of the present document are subject to continuing work within the TSG and may change following formal TSG approval. Should the TSG modify the contents of the present document, it will be re-released by the TSG with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

- x the first digit:
  - 1 presented to TSG for information;
  - 2 presented to TSG for approval;
  - 3 or greater indicates TSG approved document under change control.
- y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.
- z the third digit is incremented when editorial only changes have been incorporated in the document.

In the present document, modal verbs have the following meanings:

shall indicates a mandatory requirement to do something

shall not indicates an interdiction (prohibition) to do something

The constructions "shall" and "shall not" are confined to the context of normative provisions, and do not appear in Technical Reports.

The constructions "must" and "must not" are not used as substitutes for "shall" and "shall not". Their use is avoided insofar as possible, and they are not used in a normative context except in a direct citation from an external, referenced, non-3GPP document, or so as to maintain continuity of style when extending or modifying the provisions of such a referenced document.

should indicates a recommendation to do something

should not indicates a recommendation not to do something

may indicates permission to do something

need not indicates permission not to do something

The construction "may not" is ambiguous and is not used in normative elements. The unambiguous constructions "might not" or "shall not" are used instead, depending upon the meaning intended.

can indicates that something is possible

cannot indicates that something is impossible

The constructions "can" and "cannot" are not substitutes for "may" and "need not".

will indicates that something is certain or expected to happen as a result of action taken by an agency the behaviour of which is outside the scope of the present document

will not indicates that something is certain or expected not to happen as a result of action taken by an agency the behaviour of which is outside the scope of the present document

might indicates a likelihood that something will happen as a result of action taken by some agency the behaviour of which is outside the scope of the present document

might not indicates a likelihood that something will not happen as a result of action taken by some agency the behaviour of which is outside the scope of the present document

In addition:

is (or any other verb in the indicative mood) indicates a statement of fact

is not (or any other negative verb in the indicative mood) indicates a statement of fact

The constructions "is" and "is not" do not indicate requirements.

1 Scope

Attached to this document is an electronic copy of the floating-point C code for the Immersive Voice and Audio Services (IVAS) Codec. This C code is the unique reference specification for a standard compliant implementation of the IVAS Codec (3GPP TS 26.253), Rendering (3GPP TS 26.254), Error Concealment of Lost Packets (3GPP TS 26.255) and Jitter Buffer Management (JBM) (3GPP TS 26.256).

Requirements for any implementation of the IVAS codec to be standard compliant are specified in 3GPP TS 26.252 (Test sequences).

2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document *in the same Release as the present document*.

[1] 3GPP TR 21.905: "Vocabulary for 3GPP Specifications".

[2] 3GPP TS 26.253: "Codec for Immersive Voice and Audio Services (IVAS); Detailed Algorithmic Description including RTP payload format and SDP parameter definitions".

[3] 3GPP TS 26.254: "Codec for Immersive Voice and Audio Services (IVAS); Rendering ".

[4] 3GPP TS 26.255: "Codec for Immersive Voice and Audio Services (IVAS); Error concealment of lost packets ".

[5] 3GPP TS 26.256: "Codec for Immersive Voice and Audio Services (IVAS); Jitter Buffer Management ".

[6] 3GPP TS 26.252: "Codec for Immersive Voice and Audio Services (IVAS); Test sequences ".

[7] IETF RFC 3550: "RTP: A Transport Protocol for Real-Time Applications".

[8] Recommendation ITU-T G.191 (03/23): "Software tools for speech and audio coding standardization".

[9] Recommendation ITU-T G.192: "A common digital parallel interface for speech standardization activities".

[10] ISO/IEC 23008-3:2015: “High efficiency coding and media delivery in heterogeneous environments — Part 3: 3D audio”

[11] ISO/IEC 23091-3:2018: “Coding-independent code points — Part 3: Audio“

[12] 3GPP TS 26.442: "Codec for Enhanced Voice Services (EVS); ANSI C code (fixed-point)".

[13] 3GPP TS 26.443: "Codec for Enhanced Voice Services (EVS); ANSI C code (floating-point)".

[14] 3GPP TS 26.452: "Codec for Enhanced Voice Services (EVS); ANSI C code; Alternative fixed-point using updated basic operators".

[15] 3GPP TS 26.114: "IP Multimedia Subsystem (IMS); Multimedia Telephony; Media handling and interaction".

[16] 3GPP TR 26.902: "Video codec performance".[17] 3GPP TS 26.251: "Codec for Immersive Voice and Audio Services (IVAS); C code (fixed-point)".

3 Definitions of terms, symbols and abbreviations

3.1 Terms

For the purposes of the present document, the terms given in TR 21.905 [1] and the following apply. A term defined in the present document takes precedence over the definition of the same term, if any, in TR 21.905 [1].

bslbf: Bit string, left bit first. Bit strings are written as a string of 1s and 0s within single quote marks, for example '1000 0001'. Blanks within a bit string are for ease of reading and have no significance.

uimsbf: Unsigned integer, most significant bit first.

vlclbf: Variable length code, left bit first, where “left” refers to the order in which the variable length codes are written.

3.2 Symbols

Void.

3.3 Abbreviations

For the purposes of the present document, the abbreviations given in TR 21.905 [1] and the following apply. An abbreviation defined in the present document takes precedence over the definition of the same abbreviation, if any, in TR 21.905 [1].

ACN	Ambisonic Channel Number
CICP	Coding-independent Code Points
CSV	Comma Separated Values
EVS	Enhanced Voice Services
FB	Fullband
FEC	Frame Erasure Concealment
HRTF	Head Related Transfer Function
ISM	Independent Stream with Metadata
IVAS	Immersive Voice and Audio Services
ISAR	Immersive Audio for Split Rendering Scenarios
JBM	Jitter Buffer Management
LFE	Low Frequency Enhancement
MASA	Metadata-Assisted Spatial Audio
MC	Multi-channel
NB	Narrowband
OBA	Object Based Audio
SBA	Scene Based Audio
SID	Silence Insertion Descriptor
SWB	Super Wideband
WB	Wideband
WMOPS	Weighted Millions of Operations Per Second

4 C code structure

This clause gives an overview of the structure of the floating-point C code and provides an overview of the contents and organization of the C code attached to the present document.

C was selected as the programming language because portability was desirable.

The IVAS fixed-point C code is available in TS 26.251 [17].

4.1 Contents of the C source code

The C code is organized as listed in Table 1:

Table 1: Source code directory structure

Directory	Description
readme.txt	information on how to compile and use
Makefile	UNIX style encoder Makefile
Workspace_msvc/	Directory for the MSVC 2017 (or newer) project files
apps/	Source code files used solely for the encoder/decoder/renderer applications; these applications make use of the libraries built from lib_com, lib_dec, lib_enc, lib_rend, and lib_util
lib_com/	Source code files used both in encoder and decoder
lib_dec/	Source code files used solely in the decoder
lib_enc/	Source code files used solely in the encoder
lib_isar/	Source code files used solely for split rendering
lib_lc3plus/	Source code files used solely for split rendering
lib_rend/	Source code files used solely in the renderer
lib_util/	Source code files solely for utility functions used by the applications
scripts/	Auxiliary scripts for the conversion of HRTFs to the HRTF filter files (clause 5.10) and generation of binary renderer config metadata format" (clause 5.14.1)

The distributed files with suffix "c" contain the source code and the files with suffix "h" are the header files. The table ROM data is contained in files named "rom_*" and “ivas_rom_*” with suffix "c".

Makefiles are provided for the platforms in which the C code has been verified (listed above). Once the software is installed, this directory will have a compiled version of the encoder (named IVAS_cod), the decoder (named IVAS_dec), the renderer (named IVAS_rend) and the split rendering post-renderer (named ISAR_post_rend). In addition, this directory will have a compiled version of the encoder with support for format switching (named IVAS_cod_fmtsw) and an example program for Ambisonics format conversion (named ambi_converter).

4.2 Program execution

The codec for Immersive Voice and Audio Services is implemented in four programs and two utility executables:

- IVAS_cod: encoder;
- IVAS_dec: decoder;
- IVAS_rend: renderer;
- ISAR_post_rend: split rendering post-renderer;
- IVAS_cod_fmtsw: encoder with support for format switching;
- ambi_converter: example program for Ambisonics format conversion.

The programs should be called like:

- IVAS_cod [encoder options] <input file> <bitstream file>;
- IVAS_dec [decoder options] <bitstream file> <output file>;
- IVAS_rend [renderer options] -i <input file> -if <input format> -o <output file> -of <output format>;
- ISAR_post_rend [post-renderer options] -i < bitstream file or input file> -if <input format> -o <output file>;

- IVAS_cod_fmtsw <format_switching_file>;
- ambi_converter <input file> <output file> <input convention> <output convention>.

The input and output files contain 16-bit linear encoded PCM samples (headerless or in WAVE format) and the bitstream file contains encoded data.

The encoder, decoder, and renderer options will be explained by running the programs without any input arguments. See the file readme.txt for more information on how to run the *IVAS_cod*, *IVAS_dec*, *IVAS_rend*, *ISAR_post_rend*, *IVAS_cod_fmtsw*, *ambi_converter* programs.

5 File Formats

This clause describes the file formats used by the encoder and decoder programs. The test sequences defined in [6] also use the file formats described here.

5.1 Audio Input/output file format

For the input files read by the encoder/renderer and output files written by the decoder/renderer the following formats are supported:

- Headerless format: 16-bit integer words per each data sample. The byte order in each word depends on the host architecture (e.g. LSB first on PCs, etc.).
- WAVE format: 16-bit little-endian integer words per each data sample.

Both the encoder and the decoder program process complete frames corresponding to multiples of 20 ms.

The encoder will pad the last frame to integer multiples of 20ms frames, i.e. n speech frames will be produced from an input file with a length between [(n-1)*20ms+1 sample; n*20ms]. The files produced by the decoder will always have a length of n*20ms.

Input/output audio shall follow configurations as specified in Table 2. Ambisonics components follow the ACN ordering where $ACN_{index} = n^2 + n + m$ for real-valued spherical harmonics components of order n and degree $m = 0, 1, \ldots$, where $n = [1, \ldots, 3]$ and $m = [-n, \ldots, n]$.

Table 2: Audio track configurations

Audio format (designator)	Number of tracks	Index	Configuration (incl. ordering)	Azimuth Range	Elevation Range
Mono (M)	1	1	M	-	-
Stereo (ST)	2	1,2	L, R	-	-
Binaural (BIN)	2	1,2	L, R	-	-
Multi-channel 5.1 (MC51)	6	1	CH_A+030_E+00	+30	0
		2	CH_A-030_E+00	-30	0
		3	CH_A+000_E+00	0	0
		4	LFE	-	-
		5	CH_A+110_E+00	+100 … +120	0 … +15
		6	CH_A-110_E+00	-100 … -120	0 ... +15
	8	1	CH_A+030_E+00	+30 ... +45	0

Audio format (designator)	Number of tracks	Index	Configuration (incl. ordering)	Azimuth Range	Elevation Range
Multi-channel 7.1 (MC71)		2	CH_A-030_E+00	-30 … -45	0
		3	CH_A+000_E+00	0	0
		4	LFE	-	-
		5	CH_A+110_E+00	+85 … +110	0
		6	CH_A-110_E+00	-85 … -110	0
		7	CH_A+135_E+00	+120 … +150	0
		8	CH_A-135_E+00	-120 … -150	0
Multi-channel 5.1+4 (MC514)	10	1	CH_A+030_E+00	+30	0
		2	CH_A-030_E+00	-30	0
		3	CH_A+000_E+00	0	0
		4	LFE	-	-
		5	CH_A+110_E+00	+100 … +120	0 … +15
		6	CH_A-110_E+00	-100 … -120	0 … +15
		7	CH_A+030_E+35	+30 … +45	+30 … +55
		8	CH_A-030_E+35	-30 … -45	+30 … +55
		9	CH_A+110_E+35	+100 … +135	+30 … +55
		10	CH_A-110_E+35	-100 … -135	+30 … +55
Multi-channel 7.1+4 (MC714)	12	1	CH_A+030_E+00	+30 … +45	0
		2	CH_A-030_E+00	-30 … -45	0
		3	CH_A+000_E+00	0	0
		4	LFE	-	-
		5	CH_A+135_E+00	+120 … +150	0
		6	CH_A-135_E+00	-120 … -150	0
		7	CH_A+090_E+00	+85 … +110	0
		8	CH_A-090_E+00	-85 … -110	0
		9	CH_A+030_E+35	+30 … +45	+30 … +55
		10	CH_A-030_E+35	-30 … -45	+30 … +55
		11	CH_A+135_E+35	+100 … +150	+30 … +55
		12	CH_A-135_E+35	-100 … -150	+30 … +55
FOA (SBA1)	4	1…4	Ambisonics components with ACN_{index} 0,1,2,3	-	-

Audio format (designator)	Number of tracks	Index	Configuration (incl. ordering)	Azimuth Range	Elevation Range
HOA$\langle O \rangle$* (SBA$\langle O \rangle$)	$(\langle O \rangle + 1)^2$	1…$(\langle O \rangle + 1)^2$	Ambisonics components with ACN_{index} 0,1, 2,… $(\langle O \rangle + 1)^2$-1	-	-
Mono objects (OBA)	1…4	1…4	Object(s) with ID 1…4	-	-
Metadata-assisted spatial audio, mono (MASA1)	1	1	M	-	-
Metadata-assisted spatial audio, stereo (MASA2)	2	1,2	L, R	-	-
Combined mono MASA and OBA	2...5	1..4 2...5	Object(s) with ID 1…4 M MASA	- -	- -
Combined stereo MASA and OBA	3...6	1..4 5,6	Object(s) with ID 1…4 L, R MASA	- -	- -
Combined HOA$\langle O \rangle$* (SBA$\langle O \rangle$) and OBA	1 ... 4 + $(\langle O \rangle + 1)^2$	1…$(\langle O \rangle + 1)^2$ + 1 ... 4	Object(s) with ID 1…4 Ambisonics components with ACN_{index} 0,1, 2,… $(\langle O \rangle + 1)^2$-1	-	-

*$\langle O \rangle$ = Ambisonics order

For Ambisonics, SN3D normalization is assumed.

The azimuth ranges are expressed in degrees; positive values rotate to the left when facing the front, i.e. counter clockwise when looking from above. The elevation ranges are expressed in degrees where positive values indicate angles above the horizontal plane.

5.2 Rate switching profile (encoder input)

The encoder program can optionally read in a rate switching profile file which specifies the encoding bitrate for each frame of the input data. The rate switching profile is a binary file, generated by 'gen-rate-profile' tool, which is part of STL 2023, as contained in ITU-T G.191 [8]. The rate switching profile contains 32-bit integer words where each word represents the encoding bitrate for each particular frame. The rate switching profile is recycled if it contains less entries than the total number of frames in the input file.

5.3 Bandwidth switching profile (encoder input)

The encoder program can optionally read in a bandwidth switching profile, which specifies the encoding bandwidth for each frame of speech processed. The file is a text file where each line contains "nb_frames B". B specifies the signal bandwidth that is one of the supported bandwidths. For IVAS operation modes, WB, SWB or FB are supported. For

EVS operation modes, NB, WB, SWB and FB are supported. "nb_frames" is an integer number of frames and specifies the duration of activation of the accompanied signal bandwidth B.

5.4 Channel-aware configuration file (encoder input and decoder output)

For the EVS operation modes, the encoder program can optionally read in a configuration file which specifies the values of FEC indicator p and FEC offset o, where FEC indicator, p: LO or HI, and FEC offset, o: 2, 3, 5, or 7 in number of frames. Each line of the configuration file contains the values of p and o separated by a space.

The channel-aware configuration file is meant to simulate channel feedback from a receiver to a sender, i.e. the decoder would generate FEC indication and FEC offset values for receiver feedback that correspond to the current transmission channel characteristics, thereby allowing optimization of the transmission by the encoder which applies the FEC offset and FEC indication when in the channel-aware mode.

5.5 Object based audio metadata file (encoder/renderer input and decoder output)

For object based audio input (including the combined formats OBA + MASA and OBA + SBA), the encoder/renderer can optionally read corresponding metadata files describing the object characteristics. For bitstreams containing object based audio, the decoder can optionally write corresponding metadata files. The metadata files for object based audio (per audio object) are files consisting of comma-separated values (CSV). Each line corresponds to 20ms audio at the renderer and consists of:

- Azimuth (floating-point, range [-180°;180°[; mandatory)
- Elevation (floating-point, range [-90°;90°]; mandatory)
- Radius (floating-point, range [0; 15.75]; optional; default: 1.0)
- Spread (floating-point, range [0; 360]; optional; default: 0.0)
- Gain (floating-point, range [0;1]; optional; default: 1.0)
- Yaw (floating-point, range [-180; 180], positive indicates left; optional; default: 0.0)
- Pitch (floating-point, range [-90; 90], positive indicates up; optional; default: 0.0)
- Non-diegetic (floating-point, range [0; 1]; optional; default: 0; if Flag is set to 1, panning gain is specified by azimuth Value between [-90,90], 90 left, -90 right, 0 center)

The columns are in the following order:

```
Azimuth,Elevation,Radius,Spread,Gain,Yaw,Pitch,Non-diegetic
```

The metadata reader accepts 1-8 values specified per line. If a value is not specified, the default value is assumed.

5.6 Metadata-assisted spatial audio (MASA) metadata file (encoder/renderer input and decoder output)

For MASA audio input (including the combined format OBA + MASA), the encoder/renderer reads MASA metadata files. For bitstreams containing MASA audio, the decoder can optionally write MASA metadata files. The Syntax of the MASA metadata files is specified in Annex A.

5.7 Parameter bitstream file (encoder output / decoder input)

The files produced by the speech/audio encoder/expected by the speech decoder contain an arbitrary number of frames in the following available formats.

5.7.1 ITU-T G.192 compliant format

SYNC_WORD	DATA_LENGTH	B1	B2	…	Bnn

The encoder/decoder support parameter bitstream files according to ITU-T G.192 [9]: Each box corresponds to one `Word16` value in the bitstream file, for a total of 2+nn words or 4+2nn bytes per frame, where nn is the number of encoded bits in the frame. Each encoded bit is represented as follows: Bit 0 = 0x007f, Bit 1 = 0x0081. The fields have the following meaning:

- `SYNC_WORD:` Word to ensure correct frame synchronization between the encoder and the decoder. It is also used to indicate the occurrences of bad frames.

  In the encoder output: (0x6b21)

  In the decoder input: Good frames (0x6b21), Bad frames (0x6b20)

- `DATA_LENGTH:` Length of the speech data. Codec mode and frame type is extracted in the decoder using this parameter

5.8 VoIP parameter bitstream file (decoder input)

Packet size	Arrival time	RTP header	G.192 format (see 5.7.1)

The fields have the following size and meaning:

- Packet size: 32-bit unsigned integer (= 12 + 2 + DATA_LENGTH).
- Arrival time: 32-bit unsigned integer in ms.
- RTP header: 96 bits (see RFC 3550 [7]), including RTP timestamp and SSRC.

5.9 JBM trace file (decoder output)

The decoder can generate a JBM trace file with the –Tracefile switch as a by-product of the decoder operation in case of JBM operation (which is triggered with the –VOIP switch on the decoder side).

The trace file is a CSV file with semi-colon as separator. The trace file starts with one header line that contains the column names in the following order:

```
rtpSeqNo;rtpTs;rcvTime;playtime;active
```

For each played out speech frame one entry is written to the trace file. The interval of the playtime values is usually 20ms, but may differ, depending on the JBM operation. Each entry is a line in the trace file that contains values as specified in Table 3.

Table 3: JBM trace file entry format

Name	Unit	Description
rtpSeqNo	1	RTP sequence number of played out speech frame. -1 if no corresponding RTP packet for the speech frame exists.
rtpTs	ms	RTP time stamp of played out speech frame. -1 if no corresponding RTP packet for the speech frame exists
rcvTime	ms	Absolute reception time of the RTP packet that corresponds to the speech frame. -1 if no corresponding RTP packet for the speech frame exists.
playtime	ms	Absolute play time (i.e. the time at which the PCM data is made available by the decoder). Can be floating-point value.
active	0 or 1	Binary entry, which is set to 1 for active speech frames (i.e. frames that are neither SID nor NO_DATA)

5.10 HRTF filter file (decoder/renderer input)

Head related filters for the binaural rendering may be provided to the decoder or the renderer by using dynamic loading of external binary file. Examples code to generate such a binary file from a set of SOFA file is provided in the folder “binauralRenderer_interface” in the “script” folder of the C source code. Please refer to the readme file of this folder and sub-folder.

The main script to is the matlab script called “generate_ivas_binauralizer_tables_from_sofa.m”. It required matlab (version >= R2017b and Signal Processing Toolbox). It also requires to first generate two executables “generate_crend_ivas_tables” and “tables_format_converter”. The process to generate these two executables is decribed in the readme file. It requires c compiler and CMake to be installed.

Running the matlab script whitout modifications will generate the binaural rom tables for the different renderers for floating code (ivas_rom_binaural_crend_head.c|h, ivas_rom_binauralRenderer.c|h, ivas_rom_TDbinauralRenderer.c|h) and fixed-point code (ivas_rom_binaural_crend_head_fx.c|h, ivas_rom_binauralRenderer_fx.c|h, ivas_rom_TDbinauralRenderer_fx.c|h). It will also generate 3 binaural binary files (ivas_binaural_48kHz.bin, ivas_binaural_32kHz.bin, ivas_binaural_16kHz.bin). These 3 binary files contain default values corresponding to the values in the rom tables. By changing the sofa file used by the matlab script you can generate custom binaural binary files. The scripts are provided as example as they may not work will all sofa files.

The decoder program should be called with option -hrtf <binary_file>. This option can be used with the output configurations BINAURAL, BINAURAL_ROOM_IR and BINAURAL_ROOM_REVERB.

The binaural binary file has a specific container format with a header and a sequence of entries.

The header of a binaural binary file is defined according to Table 3A as follows:

Table 3A: Binary file header

Offset	Format	Length (in bytes)	Description
0	string	8	File identifier: “IVASHRTF”
8	integer	4	Size of file in bytes (header of file included)
12	integer	2	Number of entries (HR filters)
14	integer	4	Max size of raw data (HR filter in binary format)

Every entry contains a header followed by the related raw data which is the binary representation of the HR filter. The binary format for the different renderers are described in tables 3B through 3F.

The header of each entry is defined as given in Table 3B:

Table 3B: Entry headers

Offset	Format	Length (in bytes)	Description
0	integer	4	Renderer type The renderer type is defined according to the enumeration RENDERER_TYPE among the following values : - HRTF_READER_RENDERER_BINAURAL_FASTCONV - HRTF_READER_RENDERER_BINAURAL_FASTCONV_ROOM - HRTF_READER_RENDERER_BINAURAL_PARAMETRIC - HRTF_READER_RENDERER_BINAURAL_OBJECTS_TD - HRTF_READER_RENDERER_BINAURAL_MIXER_CONV - HRTF_READER_RENDERER_BINAURAL_MIXER_CONV_ROOM - HRTF_READER_RENDERER_REVERB_ALL
4	integer	4	Input audio configuration The input audio configuration is defined according to the enumeration BINAURAL_INPUT_AUDIO_CONFIG among the following values : - BINAURAL_INPUT_AUDIO_CONFIG_COMBINED - BINAURAL_INPUT_AUDIO_CONFIG_HOA3 - BINAURAL_INPUT_AUDIO_CONFIG_HOA2 - BINAURAL_INPUT_AUDIO_CONFIG_FOA - BINAURAL_INPUT_AUDIO_CONFIG_UNDEFINED
8	integer	4	Sampling frequency (16000, 32000, 48000)
12	integer	4	Raw data size in bytes

The format of the raw data depends on the rendering and the HR filters are represented in fix point.

Note:

- The HR filters for the renderer types HRTF_RENDERER_BINAURAL_PARAMETRIC, HRTF_RENDERER_BINAURAL_FASTCONV and HRTF_RENDERER_BINAURAL_FASTCONV_ROOM are fully defined at 48kHz.
- For the renderer type HRTF_RENDERER_BINAURAL_OBJECTS_TD the input audio configuration is always BINAURAL_INPUT_AUDIO_CONFIG_UNDEFINED.
- renderer type HRTF_READER_RENDERER_REVERB_ALL should be associated with HRTF_READER_RENDERER_BINAURAL_OBJECTS_TD and/or HRTF_READER_RENDERER_BINAURAL_CREND to specify the binaural reverberation parameters jointly with new HRIR parameters. They shall be computed on the same HRIR set.
- The binary file does not have to contain all data (HR filter) for all renderers. The following minimal configurations are accepted or any combination of those:

HRTF_READER_RENDERER_BINAURAL_FASTCONV	BINAURAL_INPUT_AUDIO_CONFIG_COMBINED	Contains data for Combined HRIR
HRTF_READER_RENDERER_BINAURAL_FASTCONV	BINAURAL_INPUT_AUDIO_CONFIG_HOA3	Contains data for HOA3
HRTF_READER_RENDERER_BINAURAL_FASTCONV	BINAURAL_INPUT_AUDIO_CONFIG_HOA2	Contains data for HOA2
HRTF_READER_RENDERER_BINAURAL_FASTCONV	BINAURAL_INPUT_AUDIO_CONFIG_FOA	Contains data for FOA
HRTF_READER_RENDERER_BINAURAL_FASTCONV_ROOM	BINAURAL_INPUT_AUDIO_CONFIG_COMBINED	Contains data for combined BRIR
HRTF_READER_RENDERER_BINAURAL_PARAMETRIC	BINAURAL_INPUT_AUDIO_CONFIG_HOA3	Contains data for HOA3, HOA2, FOA and reverberation from BRIR
HRTF_READER_RENDERER_BINAURAL_OBJECTS_TD	BINAURAL_INPUT_AUDIO_CONFIG_UNDEFINED	Contains data for HRIR
HRTF_READER_RENDERER_BINAURAL_CREND	BINAURAL_INPUT_AUDIO_CONFIG_COMBINED	Contains data for combined HRIR
HRTF_READER_RENDERER_BINAURAL_CREND	BINAURAL_INPUT_AUDIO_CONFIG_HOA3	Contains data for HOA3
HRTF_READER_RENDERER_BINAURAL_CREND	BINAURAL_INPUT_AUDIO_CONFIG_HOA2	Contains data for HOA2
HRTF_READER_RENDERER_BINAURAL_CREND	BINAURAL_INPUT_AUDIO_CONFIG_FOA	Contains data for FOA
HRTF_READER_RENDERER_BINAURAL_CREND_ROOM	BINAURAL_INPUT_AUDIO_CONFIG_COMBINED	Contains data for combined BRIR only (BINAURAL_ROOM_IR)
HRTF_READER_RENDERER_REVERB_ALL		Contains data for HRIR with reverberation (BINAURAL_ROOM_REVERB)when TD renderer or mixerconv are used

Table 3C: HR filters for binaural renderer Fastconv Impulse response binary entries

Offset	Format	Length (in bytes)	Description
0	integer	2	Scaling factor for latency value
2	integer	4	Latency value*
6	integer	2	Number of Binaural convolution bands (Nb)
8	integer	2	Number of channels (Nc)
10	integer	2	Number of taps per filter (Nt)
12	integer	2	Scaling factor for filters taps
14	integers	2 * Nb * Nc * Nt	Left ear real taps values*
14 + 2 * Nb * Nc * Nt	integers	2 * Nb * Nc * Nt	Left ear imaginary taps values*
14 + 2 * 2 * Nb * Nc * Nt	integers	2 * Nb * Nc * Nt	Right ear real taps values*
14 + 3 * 2 * Nb * Nc * Nt	integers	2 * Nb * Nc * Nt	Right ear imaginary taps values*

Table 3D: HR filters for binaural renderer Fastconv Room Impulse Response binary entries

Offset	Format	Length (in bytes)	Description
0	integer	2	Scaling factor for latency value
2	integer	4	Latency value*
6	integer	2	Number of Binaural convolution bands (Nb)
8	integer	2	Number of channels (Nc)
10	integer	2	Number of taps per filter (Nt)
12	integer	2	Scaling factor for filters taps
14	integers	2 * Nb * Nc * Nt	Left ear real taps values*
14 + 2 * Nb * Nc * Nt	integers	2 * Nb * Nc * Nt	Left ear imaginary taps values*
14 + 2 * 2 * Nb * Nc * Nt	integers	2 * Nb * Nc * Nt	Right ear real taps values*
14 + 3 * 2 * Nb * Nc * Nt	integers	2 * Nb * Nc * Nt	Right ear imaginary taps values*
14 + 4 * 2 * Nb * Nc * Nt	integer	2	CLDFB max number of channels (Nm)
16 + 4 * 2 * Nb * Nc * Nt	integer	2	Scaling factor for reverberation time values
18 + 4 * 4 * Nb * Nc * Nt	integers	2 * Nm	reverberation time values*
18 + 4 * 4 * Nb * Nc * Nt + 2 * Nm	integer	2	Scaling factor for energies corrections values
20 + 4 * 4 * Nb * Nc * Nt + 2 * Nm	integers	2 * Nm	Energies corrections values *

Table 3E: HR filters for binaural renderer parametric

Offset	Format	Length (in bytes)	Description
0	integer	2	Number of channels (Nc)
2	integer	2	Number of bins (Nb)
4	integer	2	Scaling factor for filters taps
6	integers	2 * 2 * Nc * Nb	Real taps values* one for each ear
6 + 2 * 2 * Nc * Nb	integers	2 * 2 * Nc * Nb	Imaginary taps values* one for each ear
6 + 2 * 2 * 2 * Nc * Nb	integer	2	Scaling factor for reverberation time values
8 + 2 * 2 * 2 * Nc * Nb	integers	2 * Nm	reverberation time values*
8 + 2 * 2 * 2 * Nc * Nb + 2 * Nm	integer	2	Scaling factor for energies corrections values
10 + 2 * 2 * 2 * Nc * Nb + 2 * Nm	integers	2 * Nm	Energies corrections values *
10 + 2 * 2 * 2 * Nc * Nb + 4 * Nm	integer	2	Scaling factor for early part energies corrections values
12 + 2 * 2 * 2 * Nc * Nb + 4 * Nm	integers	2 * Nm	Early part energies corrections values *

Table 3F: HR filters for binaural renderer Crend entries

Offset	Format	Length (in bytes)	Description
0	integer	2	Scaling factor for latency value
2	integer	4	Latency value*
6	integer	2	Number of HRIR/BRIR (Nc)
8	integer	2	Number of Binaural channels (Nb = 2)
10	integer	2	Max number of block iterations (Ni)
12	integers	2 * Nc * Nb	Number of iteration per channel
12 + 2 * Nc * Nb	integer	2 * Nc * Nb * Ni	Max frequency value for each block of direct part (Nf[c][b][i] Tri dimensional tab of size [Nc][Nb][Ni])
12 + 2 * Nc * Nb + 2 * Nc * Nb * Ni	integers	2	Max number of iterations for diffuse part (Nid)
14 + 2 * Nc * Nb + 2 * Nc * Nb * Ni	integers	2 * Nb	Number of diffuse iterations per binaural channel
14 + 2 * Nc * Nb + 2 * Nc * Nb * Ni + 2 * Nb	integers	2 * Nb * Nid	Max frequency value for each block of diffuse part (Nfdiff[b][i] Twoi dimensional tab of size [Nb][Ni])
14 + 2 * Nc * Nb + 2 * Nc * Nb * Ni + 2 * Nb + 2 * Nb * Nid	integer	2	Max frequency value over all diffuse blocks
16 + 2 * Nc * Nb + 2 * Nc * Nb * Ni + 2 * Nb + 2 * Nb * Nid	integer	2	Scaling factor for inverse diffuse weight values
18 + 2 * Nc * Nb + 2 * Nc * Nb * Ni + 2 * Nb + 2 * Nb * Nid	integers	2 * Nc	Left ear inverse diffuse weight values*
18 + 2 * Nc * Nb + 2 * Nc * Nb * Ni + 2 * Nb + 2 * Nb * Nid + 2 * Nc	integers	2 * Nc	Right ear inverse diffuse weight values*
18 + 2 * Nc * Nb + 2 * Nc * Nb * Ni + 2 * Nb + 2 * Nb * Nid + 2 * 2 * Nc	integer	4	Max number of bins over all HRIR/BRIR for direct part (Nbin = $\sum_{c=1,b=1,i=1}^{Nc,Nb,Ni} \mathrm{Nf[c][b][i]}$)
22 + 2 * Nc * Nb + 2 * Nc * Nb * Ni + 2 * Nb + 2 * Nb * Nid + 2 * 2 * Nc	integer	2	Scaling factor for filters taps
24 + 2 * Nc * Nb + 2 * Nc * Nb * Ni + 2 * Nb + 2 * Nb * Nid + 2 * 2 * Nc	integers	2 * Nbin	Direct part real taps values*
24 + 2 * Nc * Nb + 2 * Nc * Nb * Ni + 2 * Nb + 2 * Nb * Nid + 2 * 2 * Nc + 2 * Nbin	integers	2 * Nbin	Direct part imaginary taps values*
24 + 2 * Nc * Nb + 2 * Nc * Nb * Ni + 2 * Nb + 2 * Nb * Nid + 2 * 2 * Nc + 2 * 2 * Nbin	integer	4	Max number of bins over all HRIR/BRIR for diffuse part (Nbindiff = $\sum_{b=1,i=1}^{Nb,Ni} \mathrm{Nfdiff[b][i]}$)
24 + 2 * Nc * Nb + 2 * Nc * Nb * Ni + 2 * Nb + 2 * Nb * Nid + 2 * 2 * Nc + 2 * 2 * Nbin	integers	2 * Nbindiff	Diffuse part Real taps values*
24 + 2 * Nc * Nb + 2 * Nc * Nb * Ni + 2 * Nb + 2 * Nb * Nid + 2 * 2 * Nc + 2 * Nbdir + 2 * Nbindiff	integers	2 * Nbindiff	Diffuse part imaginary taps values*

Table 3G: HR filters for binaural renderer TD renderer entries

Offset	Format	Length (in bytes)	Description
0	integer	2	Filter method TDREND_HRFILT_Method_BSplineModel (0)
2	integer	2	Scaling factor for latency value
4	integer	4	Latency value
8	integer	2	Flag *UseItdModel* to indicate if ITD model is used
10	integer	2	Sampling rate in kHz (16, 32 or 48)
12	integer	2	Filter length, *K**
14	integer	2	Number of elevation basis functions, *P**
16	integer	2	Scaling factor for elevation knot sequence elevKSeq
18	integers	2 * (*P* - 2) = 2 * *P* - 4	Elevation knot sequence elevKSeq, length *P*-2.
14 + 2 * *P*	integers	$2\sum_{p=0}^{P-1}(3+Q_p+1) = 8*P+2\sum_{p=0}^{P-1}Q_p = 8*P+2*AlphaN$	For each elevation i: - Number of azimuth basis functions at elevation knot point *p*, $\boldsymbol{Q_p}$* - Azimuth start index - Scaling factor for azimuth knot sequence *azimKSeq* - Azimuth knot sequence *azimKSeq* of length $\boldsymbol{Q_p}+1$ (both 0 and 360 degrees included)
14 + 10 * *P* + 2 * *AlphaN*	integer	2	Number of alpha weights *AlphaN*
16 + 10 * *P* + 2 * *AlphaN*	integer	2	Scaling factor for alpha weights
18 + 10 * *P* + 2 * *AlphaN*	integers	2 * *AlphaN* * *K*	Alpha weights for left channel
18 + 10 * *P* + 2 * *AlphaN* * (1+*K*)	integers	2 * *AlphaN* * *K*	Alpha weights for right channel
18 + 10 * *P* + 2 * *AlphaN* * (1+2**K*)	integer	2	Number of unique azimuth B-spline shapes, *N_azim* Note: The B-spline shape may be recycled for several elevation knot points *p* using the azimuth shape indices and possibly a subsampling with the azimuth shape sampling factor.
20 + 10 * *P* + 2 * *AlphaN* * (1+2**K*)	integers	$2\sum_{i=0}^{N\_azim-1}(3+N\_azimBsShape[i]) = 6*N\_azim+2\sum_{i=0}^{N\_azim-1}(N\_azimBsShape[i]) = 6*N\_azim+2*azimBsTot$	For each unique azimuth B-spline shape i: - Number of azimuth B-spline shape elements, *N_azimBsShape*[*i*] - Scaling factor for azimuth B-spline shape elements - Azimuth B-spline shape elements of length *N_azimBsShape*[i] - Number of azimuth segment samples
20 + 10 * *P* + 2 * *AlphaN* * (1+2**K*) + 6 * *N_azim* + 2 * *azimBsTot*	integers	2 * *P*	Azimuth shape indices, pointing out the shape to use for each elevation knot point *p*
20 + 12 * *P* + 2 * *AlphaN* * (1+2**K*) + 6 * *N_azim* + 2 * *azimBsTot*	integers	2 * *P*	Azimuth shape sampling factor, the sampling factor to use in the shape lookup table for elevation knot point *p.*
20 + 14 * *P* + 2 * *AlphaN* * (1+2**K*) + 6 * *N_azim* + 2 * *azimBsTot*	integers	2 * 4	Elevation B-spline lengths of length HRTF_MODEL_BSPLINE_NUM_COEFFS = 4
28 + 14 * *P* + 2 * *AlphaN* * (1+2**K*) + 6 * *N_azim* + 2 * *azimBsTot*	integers	2 * 4	Elevation B-spline start indices of length HRTF_MODEL_BSPLINE_NUM_COEFFS = 4

36 + 14 * *P* + 2 * *AlphaN* * (1+2**K*) + 6 * *N_azim* + 2 * *azimBsTot*	integer	2	Number of elevation spline shape elements, *N_elev*
38 + 14 * *P* + 2 * *AlphaN* * (1+2**K*) + 6 * *N_azim* + 2 * *azimBsTot*	integer	2	Scaling factor for elevation spline shape elements
40 + 14 * *P* + 2 * *AlphaN* * (1+2**K*) + 6 * *N_azim* + 2 * *azimBsTot*	integers	2 * *N_elev*	Elevation spline shape elements of length *N_elev*
40 + 14 * *P* + 2 * *AlphaN* * (1+2**K*) + 6 * *N_azim* + 2 * *azimBsTot* + 2 * *N_elev*	integer	2	Number of elevation segment samples
42 + 14 * *P* + 2 * *AlphaN* * (1+2**K*) + 6 * *N_azim* + 2 * *azimBsTot* + 2 * *N_elev*	integer	2	Scaling factor for energy of impulse response
44 + 14 * *P* + 2 * *AlphaN* * (1+2**K*) + 6 * *N_azim* + 2 * *azimBsTot* + 2 * *N_elev*	integer	4 * 3 * *AlphaN*	Energy of impulse response, left channel, length *AlphaN* * HRTF_MODEL_N_SECTIONS = 3 * *AlphaN*
44 + 14 * *P* + 2 * *AlphaN* * (7+2**K*) + 6 * *N_azim* + 2 * *azimBsTot* + 2 * *N_elev*	integer	4 * 3 * *AlphaN*	Energy of impulse response, right channel, length 3 * *AlphaN*
If UseItdModel is 1, the following parameters are part of the binary file			
44 + 14 * *P* + 2 * *AlphaN* * (13+2**K*) + 6 * *N_azim* + 2 * *azimBsTot* + 2 * *N_elev*	integer	2	Number of elevations in ITD model, P_{ITD}
46 + 14 * *P* + 2 * *AlphaN* * (13+2**K*) + 6 * *N_azim* + 2 * *azimBsTot* + 2 * *N_elev*	integer	2	Scaling factor for ITD elevation knot sequence *elevKSeqITD*
48 + 14 * *P* + 2 * *AlphaN* * (13+2**K*) + 6 * *N_azim* + 2 * *azimBsTot* + 2 * *N_elev*	integers	2 * (P_{ITD} - 2) = 2 * P_{ITD} - 4	ITD elevation knot sequence *elevKSeqITD* in degrees
44 + 14 * *P* + 2 * *AlphaN* * (13+2**K*) + 6 * *N_azim* + 2 * *azimBsTot* + 2 * *N_elev* + 2 * P_{ITD}	integer	2	Number of ITD azimuth basis functions, Q_{ITD} Note: only one azimuth knot sequence for all elevation knot points for ITD model.
46 + 14 * *P* + 2 * *AlphaN* * (13+2**K*) + 6 * *N_azim* + 2 * *azimBsTot* + 2 * *N_elev* + 2 * P_{ITD}	integer	2	Scaling factor for azimuth knot sequence azimKSeqITD
48 + 14 * *P* + 2 * *AlphaN* * (13+2**K*) + 6 * *N_azim* + 2 * *azimBsTot* + 2 * *N_elev* + 2 * P_{ITD}	integers	2 * ((Q_{ITD} + 1) / 2 - 2) = Q_{ITD} - 3	Azimuth knot sequence azimKSeqITD, length (Q_{ITD} + 1) / 2 - 2
45 + 14 * *P* + 2 * *AlphaN* * (13+2**K*) + 6 * *N_azim* + 2 * *azimBsTot* + 2 * *N_elev* + 2 * P_{ITD} + Q_{ITD}	integer	2	Number of elements of weight *W*, *WN*
47 + 14 * *P* + 2 * *AlphaN* * (13+2**K*) + 6 * *N_azim* + 2 * *azimBsTot* + 2 * *N_elev* + 2 * P_{ITD} + Q_{ITD}	integer	2	Scaling factor for *W*
49 + 14 * *P* + 2 * *AlphaN* * (13+2**K*) + 6 * *N_azim* + 2 * *azimBsTot* + 2 * *N_elev* + 2 * P_{ITD} + Q_{ITD}	integers	2 * *WN*	Elements of *W*

49 + 14 * *P* + 2 * *AlphaN* * (13+2**K*) + 6 * *N_azim* + 2 * *azimBsTot* + 2 * *N_elev* + 2 * P_{ITD} + Q_{ITD} + 2 * *WN*	integers	2 * 4	Azimuth ITD B-spline lengths, length HRTF_MODEL_N_SECTIONS = 4
57 + 14 * *P* + 2 * *AlphaN* * (13+2**K*) + 6 * *N_azim* + 2 * *azimBsTot* + 2 * *N_elev* + 2 * P_{ITD} + Q_{ITD} + 2 * *WN*	integers	2 * 4	Azimuth ITD B-spline start indices, length HRTF_MODEL_N_SECTIONS = 4
65 + 14 * *P* + 2 * *AlphaN* * (13+2**K*) + 6 * *N_azim* + 2 * *azimBsTot* + 2 * *N_elev* + 2 * P_{ITD} + Q_{ITD} + 2 * *WN*	integer	2	Number of ITD azimuth B-spline shape elements, *N _azim_ITD*
67 + 14 * *P* + 2 * *AlphaN* * (13+2**K*) + 6 * *N_azim* + 2 * *azimBsTot* + 2 * *N_elev* + 2 * P_{ITD} + Q_{ITD} + 2 * *WN*	integer	2	Scaling factor for ITD azimuth B-spline shape elements
69 + 14 * *P* + 2 * *AlphaN* * (13+2**K*) + 6 * *N_azim* + 2 * *azimBsTot* + 2 * *N_elev* + 2 * P_{ITD} + Q_{ITD} + 2 * *WN*	integers	2 * *N_azim_ITD*	ITD azimuth B-spline shape elements, length *N_azim_ITD*
69 + 14 * *P* + 2 * *AlphaN* * (13+2**K*) + 6 * *N_azim* + 2 * *azimBsTot* + 2 * *N_elev* + 2 * P_{ITD} + Q_{ITD} + 2 * *WN* + 2 * *N_azim_ITD*	integer	2	Number of ITD azimuth segment samples
71 + 14 * *P* + 2 * *AlphaN* * (13+2**K*) + 6 * *N_azim* + 2 * *azimBsTot* + 2 * *N_elev* + 2 * P_{ITD} + Q_{ITD} + 2 * *WN* + 2 * *N_azim_ITD*	integers	2 * 4	ITD elevation B-spline lengths of length HRTF_MODEL_BSPLINE_NUM_COEFFS = 4
79 + 14 * *P* + 2 * *AlphaN* * (13+2**K*) + 6 * *N_azim* + 2 * *azimBsTot* + 2 * *N_elev* + 2 * P_{ITD} + Q_{ITD} + 2 * *WN* + 2 * *N_azim_ITD*	integers	2 * 4	ITD elevation B-spline start indices of length HRTF_MODEL_BSPLINE_NUM_COEFFS = 4
87 + 14 * *P* + 2 * *AlphaN* * (13+2**K*) + 6 * *N_azim* + 2 * *azimBsTot* + 2 * *N_elev* + 2 * P_{ITD} + Q_{ITD} + 2 * *WN* + 2 * *N_azim_ITD*	integer	2	Number of ITD elevation B-spline shape elements, *N_elev_ITD*
89 + 14 * *P* + 2 * *AlphaN* * (13+2**K*) + 6 * *N_azim* + 2 * *azimBsTot* + 2 * *N_elev* + 2 * P_{ITD} + Q_{ITD} + 2 * *WN* + 2 * *N_azim_ITD*	integer	2	Scaling factor for ITD elevation B-spline shape elements
91 + 14 * *P* + 2 * *AlphaN* * (13+2**K*) + 6 * *N_azim* + 2 * *azimBsTot* + 2 * *N_elev* + 2 * P_{ITD} + Q_{ITD} + 2 * *WN* + 2 * *N_azim_ITD*	integers	2 * *N_elev_ITD*	ITD elevation B-spline shape elements, length *N_elev_ITD*
91 + 14 * *P* + 2 * *AlphaN* * (13+2**K*) + 6 * *N_azim* + 2 * *azimBsTot* + 2 * *N_elev* + 2 * P_{ITD} + Q_{ITD} + 2 * *WN* + 2 * *N_azim_ITD* + 2 * *N_elev_ITD*	integer	2	Number of ITD elevation segment samples

* Notation refers to equation (7.2-6) in clause 7.2.2.2.2 of [2].

5.11 Head rotation trajectory file (decoder/renderer input)

In the reference implementation of the codec, input data representing the current rotation of the listeners head can be provided to the decoder in an ASCII formatted file comprising four columns separated by commas. These columns contain floating-point numbers representing either a quaternion or a Euler angle. The distinction between these two input formats is made by a magic number in the first column. If this value is set to -3.0, it is assumed that the remaining three columns contain three Euler angles. Otherwise, all four columns are interpreted as a Quaternion. The input is expected to have one line for each subframe of 5 ms.

In the case of Quaternion-based input, the columns are the w, x, y, z components of a unit quaternion. Proper normalization to 1 shall be maintained in the input. The coordinate system is defined such that the x-axis points into the direction of view, the y axis points towards the left ear, and the z axis point from bottom to top. The origin is in the center of the head. See also TS 26.253, clause 7.4.3 [2]. For example, an approximate 90-degree rotation around the vertical (z) axis would be represented by the following input line:

0.707107,0.000000,0.000000,0.70710

In the case of Euler-angle input, the first column contains the magic number -3.0, and the next three columns are the Euler angles yaw, pitch, and roll. The rotations are applied in the order yaw-pitch-roll. The yaw angle rotates around the z axis. The pitch angle rotates around the new y axis. The roll angle rotates around the new x axis. The equivalent of the example line above is then:

-3.0,90.000035,0.000000,0.000000

In case of 6 DoF support in the renderer, the head rotation trajectory file may also include a listener position in absolute Cartesian coordinates on the x-, y- and z-axis. Note that the listener position is expressed in absolute coordinates, while the listener orientation is expressed as scene displacement. An example of a listener positioned at x=3.0, y=4.0 and z=0 would be:

-3.0,90.000035,0.000000,0.000000,3.0,4.0,0.0

Note that the listener position applies for listener orientation expressed both in Quaternions and Euler angles.

5.12 Reference rotation/vector file (decoder/renderer input)

The external reference orientation of the orientation tracking feature can either be provided as a rotation (Quaternion or Euler angles) or as a pair of 3-dimensional positions (listener position and acoustic reference position).

5.12.1 Reference Rotation format

The format is identical to the format used for Head rotation trajectory file (see clause 5.11). When the rotation applied is the identity operator, the reference position is in front of the listener. Example values:

The Quaternion value “1, 0, 0, 0” places the acoustic reference **in front** of the listener, e.g. an object with azimuth 0 and elevation 0, would get rendered **in front** of the listener.

The Quaternion value “0.71, 0, 0, 0.71" (see the example in clause 5.11) places the acoustic reference **90 degrees to the right** of the listener, e.g. an object with azimuth 0 and elevation 0, would get rendered **90 degrees to the right** of the listener:

5.12.2 Reference Vector format

The Reference Vector file format describes a pair of x/y/z positions, one for the listener and one for the acoustic reference. The acoustic reference direction is defined by the vector from the listener towards the acoustic reference position.

The reference vector file is a CSV file with comma as separator. Each line shall contain a listener and an acoustic reference position in the following order:

$x_{listener}$, $y_{listener}$, $z_{listener}$, $x_{reference}$, $y_{reference}$, $z_{reference}$

Table 4: Reference Vector entry format

Name	Unit	Description
$x_{listener}$	m	x axis position of the listener.
$y_{listener}$	m	y axis position of the listener.
$z_{listener}$	m	z axis position of the listener.
$x_{reference}$	m	x axis position of the acoustic reference.
$y_{reference}$	m	y axis position of the acoustic reference.
$z_{reference}$	m	z axis position of the acoustic reference.

Example values:

The value “0, 0, 0, 1, 0, 0” places the acoustic reference **in front** of the listener, e.g., an object with azimuth 0 and elevation 0, would get rendered **in front** of the listener.

The value “0, 0, 0, -1, 0, 0” places the acoustic reference **behind** the listener, e.g., an object with azimuth 0 and elevation 0, would get rendered **behind** the listener.

The value “0, 0, 0, 1, 1, 0” places the acoustic reference **45 degrees to the right** of the listener, e.g., an object with azimuth 0 and elevation 0, would get rendered **45 degrees to the right** of the listener.

5.13 External orientation file (decoder/renderer input)

The external orientation file provides orientation information for any non-listener dependent orientations. The orientations shall be given as floating-point quaternions to the decoder/renderer in (w, x, y, z) order. Additional information may be given as HeadRotIndicator, ExtOriIndicator, ExtIntrpFlag and ExtIntrpNFrames. These options are presented in Table 5. Each entry line represents a sub-frame entry, where the sub-frame resolution is 5ms (i.e., 4 sub-frames result in a 20-ms frame).

Quaternion_W, Quaternion_X, Quaternion_Y and Quaternion_Z represent the external orientation in quaternions. The quaternion input follows the same convention as in the case of head rotations (subclause 5.11), The quaternion components shall always be present in the entry line of an external orientation file.HeadRotIndicator indicates how the head rotation is handled in the decoder/renderer. Permissive values are 0, 1, and 2. Value 0 disables the head rotation for the current sub-frame. Value 1 enables the head rotation for the current sub-frame. Value 2 freezes the head rotation value to the current head rotation. Subsequent entries with HeadRotIndicator=2 use the same head rotation as in the first entry with HeadRotIndicator=2. If HeadRotIndicator is not present in the external orientation file, a default value of 1 is used, i.e., head-tracking is applied by default according to subclause 5.11.

ExtOriIndicator indicates how the external orientation is handled in the decoder/renderer. Permissive values are 0, 1, and 2. Value 0 disables the external orientation for the current sub-frame. Value 1 enables the external orientation for the current sub-frame. Value 2 freezes the external orientation value to the current external orientation. Subsequent entries with ExtOriIndicator =2 use the same external orientation as in the first entry with ExtOriIndicator =2. If ExtOriIndicator is not present in the external orientation file, a default value of 1 is used.

ExtIntrpFlag is used to enable (value 1) or disable (value 0) interpolation for external orientations. The interpolation process interpolates to the target external orientation from the current external orientation. The target external orientation is the external orientation entry with ExtIntrpFlag=1 included in the entry. The target orientation is reached in N number of frames, where N is determined by ExtIntrpNFrames entry. If the value of ExtIntrpNFrames exceeds the maximum value of 500, the processing uses the value of 500 as the frame count for the external orientation interpolation. If ExtIntrpFlag is not present in the external orientation file, a default value of 0 is used. If ExtIntrpNFrames is not present in the external orientation file, a default value of 0 is used.

The external orientation file is an ASCII formatted file comprising input values separated by commas (i.e., a CSV file). Each line shall contain the orientation for a sub-frame in (w, x, y, z) order. Each line may also have additional entries in the following order:

Quaternion_W, Quaternion_X, Quaternion_Y, Quaternion_Z, HeadRotIndicator

OR

Quaternion_W, Quaternion_X, Quaternion_Y, Quaternion_Z, HeadRotIndicator, ExtOriIndicator

OR

Quaternion_W, Quaternion_X, Quaternion_Y, Quaternion_Z, HeadRotIndicator, ExtOriIndicator, ExtIntrpFlag

OR

Quaternion_W, Quaternion_X, Quaternion_Y, Quaternion_Z, HeadRotIndicator, ExtOriIndicator, ExtIntrpFlag, ExtIntrpNFrames

The order of the entries shall not change, and the optional entries shall not be included without first including the previous entries. For example, ExtOriIndicator shall not be contained in an entry line without first containing HeadRotIndicator.

The decoder/renderer operation is activated using option -exof <external_orientation_file>.

Table 5: External orientation entry format

Name	Format	Description	Default value	Permissive values
Quaternion_W	float	Quaternion basis element W	-	-1.0 ... 1.0
Quaternion_X	float	Quaternion basis element X	-	-1.0 ... 1.0
Quaternion_Y	float	Quaternion basis element Y	-	-1.0 ... 1.0
Quaternion_Z	float	Quaternion basis element Z	-	-1.0 ... 1.0
HeadRotIndicator	float	Indication how to handle head rotations (optional)	1	0, 1, 2
ExtOriIndicator	float	Indication how to handle external orientations (optional)	1	0, 1, 2
ExtIntrpFlag	float	Flag to enable/disable external orientation interpolation (optional)	0	0, 1
ExtIntrpNFrames	float	Number of frames to the external orientation interpolation target (optional)	0	0 – 500

Example usage:

The value “0.7, 0.7, 0, 0” applies the corresponding external orientation in the processing.

The value “0.7, 0.7, 0, 0, 0, 1” applies only the corresponding external orientation in the processing and disables the head rotation.

The value “0.7, 0.7, 0, 0, 1, 1, 1, 20” interpolates to the corresponding external orientation from the current external orientation in the span of 20 processing frames. For example, if the current external orientation is identity (1, 0, 0, 0), the external orientation is interpolated from identity to the target input orientation (0.7, 0.7, 0, 0) and the target input orientation is reached after 20 processing frames have passed.

The value “0.7, 0.7, 0, 0, 2, 1” applies the corresponding external orientation in the processing and freezes the head orientation. For example, if the current head rotation is (0.7, -0.7, 0, 0), and the next external orientation entry is “0.65, 0.75, 0, 0, 2, 1”, the next processing sub-frame uses the frozen head orientation value (0.7, -0.7, 0, 0).

5.14 Renderer config file (decoder/renderer input)

The renderer configuration file provides metadata for controlling the rendering process. This metadata includes acoustics environment parameters and source directivity. The data can be provided using binary bitstream or a text file. The binary bitstream format is intended to be used while providing rendering configuration remotely, e.g., associated with audio content as distributed by a content provider. The text format is intended to be used locally on the UE. The binary configuration bitstream is provided from a file. A path to the binary bitstream file is provided in the text configuration file.

5.14.1 Binary renderer config metadata format

The syntax of the binary renderer config metadata format is specified in Annex B.

5.14.2 Text renderer config metadata format

The text based renderer configuration file contains the following syntax elements:

[general]	header of general metadata
binaryConfig = path;	path to the binary configuration file
[roomAcoustics]	header of room acoustic metadata group
frequencyGridCount = N;	number of frequency grids
acousticEnvironmentCount = N;	number of acoustic environments
[frequencyGrid:N]	header of a frequency grid, where N is a zero-based, sequential grid index
method = individualFrequencies | startHopAmount | defaultBanding;	specifies frequency grid representation method
nrBands = N;	number of frequency bands, applicable for individual frequencies and start-hop-amount representation methods
frequencies = [...];	center frequencies for individualFrequencies representation method, a comma separated list of N numeric values (ints or floats)
startFrequency = value;	starting frequency for start-hop-amount representation method
frequencyHop = value;	frequency hop for start-hop-amount representation method. Center frequencies for a grid are computed as $fc_n = fc_{n-1} * hop$
defaultGrid = N;	default grid identifier. The available default grids are as in Annex B.1, Table B.4.
defaultGridOffset = N;	it is possible to use a subset of a default grid by specifying an offset - index of the first center frequency of the default grid and
defaultGridNrBands = N;	number of bands from the default grid to be used

[acousticEnvironment:N]	header of an acoustic environment element, where N is a zero-based grid index (does not have to be sequential)
frequencyGridIndex = N;	index of the frequency grid (see above) used for frequency dependent parameters
preDelay = value;	a delay at which DSR (diffuse to source ratios) were measured
rt60 = [...];	RT60 values per frequency band
dsr = [...];	diffuse to source sound energy ratio per frequency band
earlyReflectionsSize = [x, y, z];	shoebox model room size in x, y, z dimension in meters
absorptionCoeffs = [x1, x2, y1, y2, z1, z2];	early reflections absorption coefficients per wall
listenerOrigin = [x, y, z];	early reflections listener origin (optional) as offset from the room center
lowComplexity = TRUE | FALSE;	early reflection low-complexity mode flag (FALSE by default)
[directivitySetting]	header of the directivity data group
directivityCount = N;	number of directivity components
[directivityPattern:N]	header of a directivity pattern element, where N is a zero-based element index
[distanceAttenuation]	header of the distance attenuation data group
maxDist = md;	Max distance for distance attenuation function
refDist = rd;	Ref (minimum) distance for distance attenuation function
rolloffFactor = rf;	Rolloff-factor for distance attenuation function
directivity = [ia, oa, og];	directivity data: ia – inner angle, oa – outer angle, og – outer gain.
[SPLITREND]	header of split rendering group
BITRATE = R;	split rendering bitrate
DOF = N;	degree of freedom (N ranging from 0 to 3)
HQMODE = N;	High quality mode for 3DOF (N can be 0 or 1), adds more complexity at pre-renderer
CODEC = X;	split rendering transport codec (X can be LCLD or LC3plus or NONE)
FRAMESIZE = [5, 10, 20]	frame size in ms of the split rendering transport codec. Note: LC3plus supports 5 and 10 ms framesize, LCLD supports 5, 10 and 20 ms framesize.

The config file format supports comments starting with a hash sign #. It also supports splitting data into multiple lines, useful in case of larger arrays.

5.15 Scene description file (renderer input)

The renderer can render scenes consisting of one or multiple sources. The scenes can be described using a scene description file (textfile) which is defined according to Table 6:

Table 6 : Scene Description File Syntax

Line no.	Type	Description
1	string	Path to a “multitrack” audio file. This shall be a single multichannel wav/pcm

<table>
<tr><td></td><td></td><td>file that contains all input audio. For example, channels 1-4 can be an FOA scene,
channel 5 – an object and channels 6-11 – a 5.1 channel bed.
The path given shall be relative to the location of the config file.
This path has lower priority than the one given on the command line: it is
ignored if the –inputAudio argument to the renderer executable is specified.</td></tr>
<tr><td>2</td><td>integer</td><td>Contains number of inputs. An input may either be an Ambisonics scene, an
object or a channel bed. This does not correspond the total number of channels in the input audio file.
The renderer simultaneously supports:
- 1 Ambisonics input
- 1 Channel-based input
- 1 MASA input
- Up to 4 audio objects (ISM) inputs</td></tr>
<tr><td>Following lines</td><td></td><td>Definition of each of the inputs. Inputs may be listed in any order. They are not required to be listed in the same order as in the audio file.
Ambisonics:
<table>
<tr><th>Line no. (relative to each input)</th><th>Type</th><th>Description</th></tr>
<tr><td>1</td><td>string</td><td>SBA</td></tr>
<tr><td>2</td><td>integer</td><td>Index of the first channel of this input in the multitrack file (1-indexed) Ambisonics order</td></tr>
</table>
Channel-based:
<table>
<tr><th>Line no. (relative to each input)</th><th>Type</th><th>Description</th></tr>
<tr><td>1</td><td>string</td><td>MC</td></tr>
<tr><td>2</td><td>integer</td><td>Index of the first channel of this input in the multitrack file (1-indexed)</td></tr>
<tr><td>3</td><td>string</td><td>Name of speaker layout (X_Y_Z or CICPx format)</td></tr>
</table>
MASA:
<table>
<tr><th>Line no. (relative to each input)</th><th>Type</th><th>Description</th></tr>
<tr><td>1</td><td>string</td><td>MASA</td></tr>
<tr><td>2</td><td>integer</td><td>Index of the first channel of this input in the multitrack file (1-indexed)</td></tr>
<tr><td>3</td><td>integer</td><td>Number of transport channels</td></tr>
<tr><td>4</td><td>string</td><td>Path to MASA metadata file, see clause 5.6. The path shall be relative to config file location.</td></tr>
</table></td></tr>
</table>

Audio Objects, Option 1:

Line no. (relative to each input)	Type	Description
1	string	`ISM`
2	integer	Index of this input's audio in the multitrack file (1-indexed)
3	string	Path to Object based audio metadata file, see clause 5.5. The path shall be relative to config file location.

or

Audio Objects, Option 2:

Line no. (relative to each input)	Type	Description
1	string	`ISM`
2	integer	Number N of positions defined, followed by N lines in form:
Following N lines:	integer, float, (…)	Number of frames for which the defined position is kept, object position according to values defined in Object based audio metadata file, see 5.5.

Each input definition may be followed by a list of optional properties in the following format:

`<property_key>:<property_value>`

Each key-value pair shall be placed on a separate line.

The following key-value pairs are supported:

Key	Type	Description
gain_dB	float	Applied gain to input in dB
lfe_matrix	string	CSV file containing a LFE panning matrix. File containing a containing a matrix of dimensions [num_input_lfe x num_output_channels] with elements specifying linear routing gain. If specified, overrides the output LFE position option and the default behavior which attempts to map input to output LFE channel(s)
lfe_gain_dB	float	Applied gain to input LFE in dB. Applicable only to formats containing an LFE.
lfe_azi	float	Azimuth defining LFE position on sphere; alternative to lfe_matrics.

lfe_ele	float	Elevation defining LFE position on sphere; alternative to lfe_matrics.

Example configuration:

The following example defines a scene with 4 inputs:

- ISM with trajectory defined in a separate file. Channel 12 in the input file. Apply a gain of 0.5 dB.
- Ambisonics, order 1. Channels 1-4 in the input audio file. Apply -6 dB of gain.
- CICP6 channel bed. Channels 5-10 in the input audio file.
- ISM with 2 defined positions (-90,0) and (90,0). Channel 11 in the input file. The object will start at position (-90,0) and stay there for 5 frames, then move to (90,0) and stay there for 5 frames. This trajectory is looped ver the duration of the input audio file.

```
./input_audio.wav
4
ISM
12
path/to/IVAS_ISM_metadata.csv
gain_dB:0.5
SBA
1
1
gain_dB:-6
MC
5
5_1
ISM
11
2
5,-90,0
5,90,0
```

5.16 Split rendering pose correction file (decoder/renderer output, post-renderer input)

The split rendering pose correction file used with PCM split rendering audio data (output of decoder/renderer and input to post-renderer, mode `BINAURAL_SPLIT_PCM`) is described in TS 26.253, clause 7.6.2.3 and clause 7.6.7.

5.17 Split rendering bitstream file (decoder/renderer output, post-renderer input)

The split rendering bitstream file (output of decoder/renderer and input to post-renderer, mode `BINAURAL_SPLIT_CODED`) is described in TS 26.253, clause 7.6.7.

5.18 Object editing file (decoder input)

For object based audio input (including the combined formats OBA + MASA and OBA + SBA), the decoder supports editing of object characteristics while decoding/rendering. This allows for a scene adjustment on receiver side.

The parameters for the object editing in decoder for the supported formats are provided via a text parameter file. Each row of the file corresponds to one 20ms IVAS frame. The row contains one or more of the following parameters separated by a comma, as described in Table 7.

Table 7: Object Editing File Parameters

Parameter	Description
`bg_gain=<float>`	linear gain to be applied on the SBA/MASA component in OSBA/OMASA, no effect for ISM
`obj_<int>_gain=<float>`	linear gain to be applied on object <int>, 0-based indexing
`obj_<int>_relgain=0|1`	if 1, obj_<int>_gain is interpreted as a relative modification. default is absolute modification
`obj_<int>_azi=<float>`	azimuth angle in degrees to be applied on object <int>, 0-based indexing
`obj_<int>_relazi=0|1`	if 1, obj_<int>_azi is interpreted as a relative modification. default is absolute modification
`obj_<int>_ele=<float>`	elevation angle in degrees to be applied on object <int>, 0-based indexing
`obj_<int>_relele=0|1`	if 1, obj_<int>_ele is interpreted as a relative modification. default is absolute modification

In addition to these metadata parameters, editing of extended metadata parameters is supported for Discrete ISM, OMASA Discrete ISM and OSBA Discrete ISM input formats. Extended metadata parameters consist of radius, yaw and pitch, and they are described in Table 8.

Table 8: Object Editing File Extended Metadata Parameters

Parameter	Description
`obj_<int>_radius=<float>`	linear radius to be applied on object <int>, 0-based indexing
`obj_<int>_relradius=0|1`	if 1, obj_<int>_radius is interpreted as a relative modification. default is absolute modification
`obj_<int>_yaw=<float>`	yaw angle in degrees to be applied on object <int>, 0-based indexing
`obj_<int>_relyaw=0|1`	if 1, obj_<int>_yaw is interpreted as a relative modification. default is absolute modification
`obj_<int>_pitch=<float>`	pitch angle in degrees to be applied on object <int>, 0-based indexing
`obj_<int>_relpitch=0|1`	if 1, obj_<int>_pitch is interpreted as a relative modification. default is absolute modification

If a parameter is not specified, that parameter is not edited. An empty line in the file corresponds to not editing any parameter in the item.

5.19 RTPDUMP file (encoder output, decoder input)

The rtpdump file format is used as the interchange format of IVAS when RTP packing or unpacking is included as part of the encoder or decoder operation, respectively.

The rtpdump file format has already been used in the EVS decoder [12, 13, 14], MTSI [15] and TR 26.902 [16]. It has been originally proposed by Henning Schulzrinne, see http://www.cs.columbia.edu/IRT/software/rtptools/. Within the scope of this IVAS, only the binary version of the file format is of relevance. The file is constructed as follows:

The file starts with one line of ASCII coded text, indicating:

```
#!rtpplay1.0 address/port\n
```

wherein "address" stands for an IP address (e.g. 192.168.1.2) and port stands for a port number, e.g. 1234. Neither value is used by the toolchain employed in this report. "\n" stands for carriage return/linefeed.

The ASCII header is followed by one binary header (RD_hdr_t) and one RD_packet_t structure for each received packet. All fields are in network byte order. The RTP and RTCP packets are recorded as-is.

```
typedef struct {
  struct timeval start;  /* start of recording (GMT) */
  u_int32 source;        /* network source (multicast address) */
  u_int16 port;          /* UDP port */
} RD_hdr_t;

typedef struct {
  u_int16 length;    /* length of packet, including this header (may
                        be smaller than plen if not whole packet recorded) */
  u_int16 plen;      /* actual header+payload length for RTP, 0 for RTCP */
  u_int32 offset;    /* milliseconds since the start of recording */
} RD_packet_t;
```

Annex A (normative): Metadata-assisted spatial audio (MASA) format

A.1 General

This Annex describes the Metadata-assisted spatial audio (MASA) format. The MASA format consists of audio signals and metadata. The audio signals for MASA can be mono or stereo. The metadata shall be provided according to a structure defined here, and it comprises descriptive metadata and spatial metadata, as defined in the following clauses.

A.2 MASA format metadata structure

MASA format input to IVAS encoder follows the 20-ms frame size. For each 20-ms audio frame, one corresponding metadata frame is provided. Each metadata frame is structured as illustrated in Figure A.1. The descriptive metadata common for the whole frame is written first. This is followed by the spatial metadata, which consists of four spatial metadata subframes, each corresponding to 5 ms of audio. The structure of the spatial metadata subframes depends on the number of direction parameters in the frame. There are two options for the structure, illustrated in Figure A.2 and Figure A.3 for one direction and two directions, respectively.

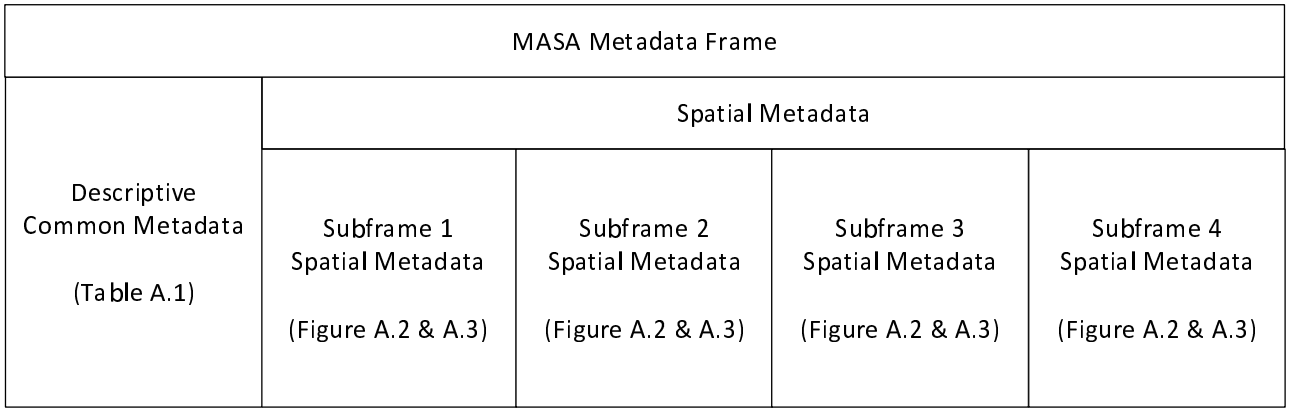

Figure A.1: Metadata structure for one MASA input signal frame

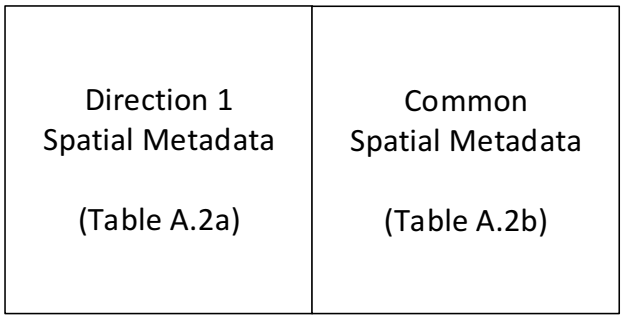

Figure A.2: MASA spatial metadata structure for one subframe with one direction

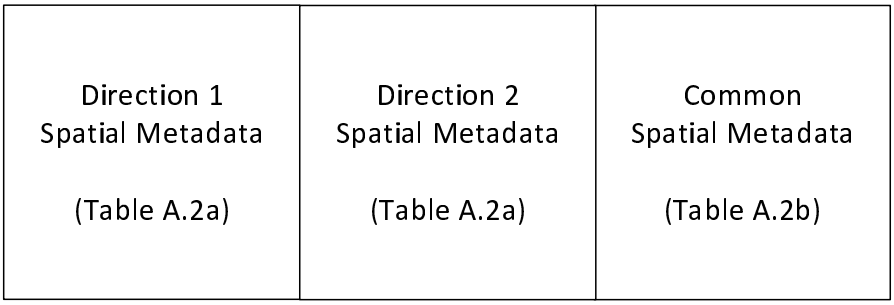

Figure A.3: MASA spatial metadata structure for one subframe with two directions

Table A.1 presents the MASA descriptive common metadata parameters in order of writing. The definitions and use of the descriptive metadata parameters are described in clause A.4.

Table A.2a and Table A.2b present the MASA spatial metadata parameters dependent and independent of the number of directions, respectively. The definitions and use of the spatial metadata parameters are described in clause A.5.

Table A.1: MASA format descriptive common metadata parameters

Field	Bits	Description
Format descriptor	64	Defines the MASA format for IVAS. Eight 8-bit ASCII characters: 01001001, 01010110, 01000001, 01010011, 01001101, 01000001, 01010011, 01000001 Values stored as 8 consecutive 8-bit unsigned integers.
Channel audio format	16	Combined following fields stored in two bytes. Value stored as a single 16-bit unsigned integer.
Number of directions	(1)	Number of directions described by the spatial metadata. Each direction is associated with a set of direction dependent spatial metadata. Range of values: [1, 2]
Number of channels	(1)	Number of transport channels in the format. Range of values: [1, 2]
Source format	(2)	Describes the original format from which MASA was created.
(Variable description)	(12)	Further description fields based on the values of ‘Number of channels’ and ‘Source format’ fields. When all bits are not used, zero padding is applied.

Table A.2a: MASA format spatial metadata parameters (dependent of number of directions)

Field	Bits	Description
Direction index	16	Direction of arrival of the sound at a time-frequency parameter interval. Spherical representation at about 1-degree accuracy. Range of values: “covers all directions at about 1° accuracy” Values stored as 16-bit unsigned integers.
Direct-to-total energy ratio	8	Energy ratio for the direction index (i.e., time-frequency subframe). Calculated as energy in direction / total energy. Range of values: [0.0, 1.0] Values stored as 8-bit unsigned integers with uniform spacing of mapped values.
Spread coherence	8	Spread of energy for the direction index (i.e., time-frequency subframe). Defines the direction to be reproduced as a point source or coherently around the direction. Range of values: [0.0, 1.0] Values stored as 8-bit unsigned integers with uniform spacing of mapped values.

Table A.2b: MASA format spatial metadata parameters (independent of number of directions)

Field	Bits	Description
Diffuse-to-total energy ratio	8	Energy ratio of non-directional sound over surrounding directions. Calculated as energy of non-directional sound / total energy. Range of values: [0.0, 1.0] (Parameter is independent of number of directions provided.) Values stored as 8-bit unsigned integers with uniform spacing of mapped values.
Surround coherence	8	Coherence of the non-directional sound over the surrounding directions. Range of values: [0.0, 1.0] (Parameter is independent of number of directions provided.) Values stored as 8-bit unsigned integers with uniform spacing of mapped values.
Remainder-to-total energy ratio	8	Energy ratio of the remainder (such as microphone noise) sound energy to fulfil requirement that sum of energy ratios is 1. Calculated as energy of remainder sound / total energy. Range of values: [0.0, 1.0] (Parameter is independent of number of directions provided.) Values stored as 8-bit unsigned integers with uniform spacing of mapped values.

A.3 MASA format time-frequency resolution

The MASA spatial metadata parameters describe the spatial characteristics of the captured spatial sound scene. The parametric representation is based on frequency bands. A certain spatial characteristic thus relates to a frequency band, and a neighbouring frequency band can exhibit a different characteristic. For MASA format, 24 frequency bands are used. Table A.3 presents these frequency bands.

The metadata frame corresponding to 20-ms frame of audio is divided into four subframes of 5 ms each, which allows for higher temporal resolution of the spatial characteristics than offered by the frame size. The parametric representation in each frame therefore consists of 24 frequency bands in 4 time slots giving a total of 96 time-frequency tiles.

When a frame describes the scene using one spatial direction, there are 96 instances of each of the spatial metadata parameters corresponding with the 96 time-frequency tiles. When a frame describes the scene using two spatial directions, there are two values per time-frequency tile for some of the spatial metadata parameters. In this case, there are 192 instances of those spatial metadata parameters in one metadata frame.

Table A.3. MASA spatial metadata frequency bands

Band	LF (Hz)	HF (Hz)	BW (Hz)
1	0	400	400
2	400	800	400
3	800	1200	400
4	1200	1600	400
5	1600	2000	400
6	2000	2400	400
7	2400	2800	400
8	2800	3200	400
9	3200	3600	400
10	3600	4000	400
11	4000	4400	400
12	4400	4800	400
13	4800	5200	400
14	5200	5600	400
15	5600	6000	400
16	6000	6400	400
17	6400	6800	400
18	6800	7200	400
19	7200	7600	400
20	7600	8000	400
21	8000	10000	2000
22	10000	12000	2000
23	12000	16000	4000
24	16000	24000	8000

A.4 MASA descriptive metadata parameters

The MASA descriptive metadata is provided once per frame. It includes information for correctly reading the metadata frame and information relating to creation of the current MASA format signal and its transport audio signals that can be used to assist encoding or rendering of the spatial audio.

The parameter fields of Table A.1 as defined as follows:

Format descriptor (64 bits)

The unique format descriptor code is provided at the beginning of every MASA format metadata frame. It specifies MASA format for the IVAS codec.

Required bit value	Decoded value	Additional description
01001001, 01010110, 01000001, 01010011, 01001101, 01000001, 01010011, 01000001	“IVASMASA”	Unique format descriptor

Channel audio format (16 bits as specified below)

Two bytes providing the following individual fields:

- Number of directions
- Number of channels
- Source format

and a variable 12-bit description configured based on ‘Number of channels’ and ‘Source format’.

Number of directions (1 bit)

This parameter field indicates how many directions are described in current MASA format frame. Size of the metadata associated with the current frame depends on the number of directions.

Bit value	Decoded value	Additional description
0	1 direction	-
1	2 directions	-

Number of channels (1 bit)

This parameter field indicates how many transport channels are used for the MASA format. This parameter is required by the codec or renderer in some form to read the correct number of channels. Some additional channel format descriptors further depend on the number of channels.

Bit value	Decoded value	Additional description
0	1 channel	-
1	2 channels	-

Source format (2 bits)

This parameter field describes the format of source signals that were used to form the MASA format input file/stream. This parameter provides additional information that can benefit encoding, decoding, and/or rendering. First bit value (00) is the default value.

Bit value	Decoded value	Additional description
00	Default/Other	Audio originates from unknown format(s) including mixed sources
01	Microphone grid	Audio originates from various (irregular) microphone grids (e.g., smartphones or other UEs)
10	Channel-based	Audio originates from premixed channel-based audio (e.g., 5.1)
11	Ambisonics	Audio originates from Ambisonics format

Variable description (12 bits including zero padding)

Based on the values of the 'Number of channels' bit and 'Source format' bits, the variable description is configured to provide up to three additional fields to further describe the source format or transport channels. This information can guide, e.g., metadata encoding and rendering. The following presents the possible field combinations and their definitions.

Source format == 00 (Default/Other)

If number of channels is 1 (bit value 0), no additional metadata is specified. Instead, 12-bit zero padding is applied.

If number of channels is 2 (bit value 1), following additional fields are configured in order:

- Transport definition field (3 bits). This field describes the configuration of the two transport channels. The possible bit values and corresponding configurations are provided in Table A.4.
- Channel angle field (3 bits). This field describes symmetric angle positions for transport signals with directivity patterns. In this notation, 0° corresponds to the front. The bit values and corresponding configuration are defined in Table A.5.
- Channel distance field (6 bits). The bit values and corresponding configuration are defined in Table A.6.

Table A.4: Transport definition field for Source formats: Default/Other and Microphone grid

Bit value	Decoded value	Additional description
000	Unknown/Other	Default
001	Omni	-
010	Subcardioid	-
011	Cardioid	-
100	Supercardioid	-
101	Hypercardioid	-
110	Dipole	-
111	Binaural	-

Table A.5: Channel angles for directive patterns for Source formats: Default/Other and Microphone grid

Bit value	Decoded value	Additional description
000	Unspecified	Default
001	±90 deg.	-
010	±70 deg.	XY stereo
011	±55 deg.	XY stereo, ORTF stereo
100	±45 deg.	NOS stereo, XY stereo, Blumlein pair
101	±30 deg.	-
110	±0 deg.	AB stereo. Needs spacing for stereo image.
111	Reserved	-

Note: If Transport definition value is “Unknown”, “Omni”, or “Binaural”, value 000 is used.

The channel distance parameter is defined with a few predefined values and the distance values between 0.01 m and 1 m are calculated as an equal multiplicative interval such that there are 60 values from 0.01 m to 1 m. The equation for this is given as:

$$d_{dec} = \frac{\left(\sqrt[59]{100}\right)^{B-3}}{100}$$

where d_{dec} is the decoded distance value and B is the bit value as an integer value, i.e., $B = 3, \ldots, 62$. The result is in meters.

Table A.6: Channel distance for Source formats: Default/Other and Microphone grid

Bit value	Decoded value	Additional description
000000	Unspecified	Distance is not specified, or it is unknown
000001	0 m / coincident	No distance between microphones, i.e., they are coincident
000010	< 0.01 m	Distances smaller than 0.01 m
000011	0.01 m	(Distances formed with equation above)
…	…	(Distances formed with equation above)
111110	1 m	(Distances formed with equation above)
111111	> 1 m	Distances larger than 1 m

Source format == 01 (Microphone grid)

If number of channels is 1 (bit value 0), no additional metadata is specified. Instead, 12-bit zero padding is applied.

If number of channels is 2 (bit value 1), following additional fields are configured in order:

- Transport definition field (3 bits). This field describes the configuration of the two transport channels. The possible bit values and corresponding configurations are provided in Table A.4.

- Channel angle field (3 bits). This field describes symmetric angle positions for transport signals with directivity patterns. In this notation, 0° corresponds to the front. The bit values and corresponding configuration are defined in Table A.5.

- Channel distance field (6 bits). The bit values and corresponding configuration are defined in Table A.6.

The field definitions used for Microphone grid source format and Default/Other source format are the same. Differentiation is based on Source format parameter itself.

Source format == 10 (Channel-based)

For premixed content, the original channel layout can be provided. In addition to common CICP layouts relevant for IVAS, two generic options (3D and 2D) are available. The description of the bit values is provided in Table A.7. The transport signals with this source format are assumed to be a mono (1 channels) or left-right stereo (2 channels) downmix of the multi-channel signals, and thus the number of channels can be 1 or 2 (bit values 0 or 1).

In addition to the 3-bit Channel layout field, 9 bits of zero padding is applied to complete the 12-bit variable description.

Table A.7: Channel layout field for the channel-based source format

Bit value	Decoded value	Additional description
000	Unknown/Other	Unknown layout or other (3D) layout. Default option.
001	Other planar	Other 2D layout
010	2.0	CICP2 positions, ITU order
011	5.1	CICP6 positions, ITU order
100	5.1+2	CICP14 positions azimuth, 35° elevation, ITU order
101	5.1+4	CICP16 positions azimuth, 35° elevation, ITU order
110	7.1	CICP12 positions, ITU order
111	7.1+4	CICP19 positions azimuth, 35° elevation, ITU order

Note 1: ITU channel order is given in ISO/IEC 23008-3:2015 [10], Table 95.
Note 2: Azimuth positions are given in ISO/IEC 23091-3:2018 [11], Table 3.

Source format == 11 (Ambisonics)

If number of channels is 1 (bit value 0), no additional metadata is specified. Instead, 12-bit zero padding is applied.

If number of channels is 2 (bit value 1), following two additional fields are configured in order:

- Transport definition field (3 bits). This describes the configuration of the two transport channels. The possible bit values and corresponding configurations are provided in Table A.4. However, bit values 001 (omni) and 111 (binaural) are not allowed and are interpreted as bit value 000.

- Channel angle field (3 bits). Describes symmetric angle positions for transports signals with directive patterns. In this notation, 0° corresponds to the front. This is defined in Table A.5.

- In addition, 6 bits of zero padding is applied to complete the 12-bit variable description.

For Ambisonics-based transport signals, transport channels are considered coincident, and there is therefore no ‘Channel distance’ field specified.

A.5 MASA spatial metadata parameters

The MASA spatial metadata describes the spatial audio characteristics corresponding to the one or two transport audio signals. Thus, the spatial audio scene can be rendered for listening based on the combination of the transport audio signals and the spatial metadata.

The MASA spatial metadata is provided once per subframe in each frame following the time-frequency resolution presented in clause A.3. Spatial metadata for each subframe contains one or two first sets of parameters depending on the number of directions (as defined by the corresponding metadata field in descriptive metadata, clause A.4) and one second set of parameters that does not depend on the number of directions. As shown in Figure A.2 and Figure A.3, the parameters corresponding to Table A.2a are written first in the stream, followed by the parameters corresponding to Table A.2b.

The definitions and use of the MASA spatial metadata parameters are described in order in the following.

Direction index: Spatial direction(s)

Spatial directions represent the directional energy flows in the sound scene. Each spatial direction together with corresponding direct-to-total energy ratio describes how much of the total energy for each time-frequency tile is coming from that specific direction. In general, this parameter can also be thought of as the direction of arrival (DOA).

There can be one or two spatial directions for each time-frequency tile in the input metadata. Each spatial direction is represented using a 16-bit direction index. This is an efficient representation of directions as points of a spherical grid with an accuracy of about 1 degree in any arbitrary direction.

The direction indexing corresponds to the function for transforming the audio direction angular values (azimuth ϕ and elevation θ) into an index, and the inverse function for transforming the index into the audio direction angular values.

Each pair of values containing the elevation and the azimuth is first quantized on a spatial spherical grid of points and the index of the corresponding point is constructed. The structure of the spherical grid is defined first, followed by the quantization function and lastly the index formation followed by the corresponding de-indexing function.

The spherical grid is defined as a succession of horizontal circles of points. The circles are distributed on the sphere, and they correspond to several elevation values. The indexing functions make the connection between the angles (elevation and azimuth) corresponding to each of these points on the grid and a 16-bit index.

The spherical grid is on a sphere of unitary radius that is defined by the following elements:

- The elevation values are equidistant between -90 and +90 degrees; the value 0 is represented and corresponds to the circle situated on the equator. The values are symmetrical with respect to the origin. The number of positive elevation values is $N_\theta = 122$.
- For each elevation value there are several equally spaced azimuth values. One point on the grid is given by the elevation and the azimuth value. The number $n(i)$ of azimuth values is calculated as follows:
  - on the equator of the spherical grid ($\theta = 0$) is it set to

$$n(1) = 430$$

  - there is one point at each of the poles ($\theta = \pm 90$ degrees)

$$n(N_\theta) = 1$$

  - the function calculating the number of points $n(i)$ on the grid for other elevation indices, $i = 2, \ldots, N_\theta - 1$, uses the following definition: $n(i) = \frac{(cumN(i) - cumN(i-1))}{2}$

  with $cumN(1) = 0$ and

$$cumN(i) = 2\ round_i\left(\frac{2^{16}-432}{2}\frac{\sin\left((i-\frac{1}{2})\delta\right)-\sin\left(\frac{\delta}{2}\right)}{\sin\left((N_\theta-\frac{3}{2})\delta\right)-\sin\left(\frac{\delta}{2}\right)}\right)$$

where δ is the uniform quantization step for $i = 1, \ldots, N_\theta - 1$, $2\ round_i(x/2)$ is a rounding function to the nearest even integer (above x for $i = 2$, closest for $i > 2$). The term $cumN(i)$ gives the cumulative cardinality (i.e., cumulative number of points in the spherical grid) in a spherical zone going from the first non-zero elevation value to the i-th elevation value. This cumulative cardinality is derived from the relative area on the spherical surface, assuming a (near) uniform point distribution of the remaining number of points $2^{16} - 432$ (let alone the equator and poles).

- The azimuth values start from the front direction and are in trigonometrical order from 0 to 2π.
- The quantized azimuth values for odd values of i are equally spaced and start at 0.
- The quantized azimuth values for even values of i are equally spaced and start at $\frac{\pi}{n(i)}$.
- There is a same number of quantized azimuth values for same absolute value elevation codewords.

The quantization in the spherical grid is done as follows:

- The elevation value is quantized in the uniform scalar quantizer to the two closest values θ_1, θ_2
- The azimuth value is quantized in the azimuth scalar quantizers corresponding to the elevation values θ_1, θ_2
- The distance on the sphere is calculated between the input elevation azimuth pair and each of the quantized pairs $(\theta_1, \phi_1), (\theta_2, \phi_2)$

$$d_i = -(\sin\theta \sin\theta_i + \cos\theta_i \cos(\phi - \phi_i)), i = 1{:}\,2$$

- The pair with lower distance is chosen as the quantized direction.

The resulting quantized direction index is obtained by enumerating the points on the spherical grid by starting with the points for null elevation first, then the points corresponding to the smallest positive elevation codeword, the points corresponding to the first negative elevation codeword, followed by the points on the following positive elevation codeword and so on.

Direct-to-total energy ratio(s)

Direct-to-total energy ratios work together with spatial directions as described above. Each direct-to-total energy ratio corresponds to a specific spatial direction and describes how much of the energy comes from that specific spatial direction compared to the total energy.

Spread coherence

Spread coherence is a parameter that describes the directional energy flow further. It represents situations where coherent directional sound energy is coming from multiple directions at the same time. This is represented with a single spread coherence parameter that describes how the sound should be synthesized.

In synthesis, this parameter should be used such that value 0 means that the sound is synthesized to single direction as directed by the spatial direction, value 0.5 means that the sound is synthesized to the spatial direction and two surrounding directions as coherent, and 1 means that the sound is synthesized to two surrounding directions around the spatial direction.

Diffuse-to-total energy

Diffuse-to-total energy ratio represents non-directional energy flow in the sound scene. This is a complement to the direct-to-total energy ratios and in an ideal capture with no undesired signal (or synthesized sound scene), the diffuse-to-total ratio value is always

$$r_{diff} = 1 - \sum r_{dir}.$$

Surround coherence

Surround coherence is a parameter that describes the non-directional energy flow. It represents how much of the non-directional energy should be presented as coherent reproduction instead of decorrelated reproduction.

Remainder-to-total energy ratio

Remainder-to-total represents all the energy that does not "belong" to the captured sound scene based on the used model. This includes possible microphone noise and other capture artefacts that have not been removed from the signal in pre-processing. This means that by considering the direct-to-total energy ratio, the diffuse-to-total energy ratio, and the remainder-to-total energy we end up with a complete energy ratio model of

$$\sum_{m}^{M} r_{dir}(m) + r_{diff} + r_{rem} = 1.$$

when there is any remainder energy present. Otherwise, the energy ratio equation defined for diffuse-to-total energy ratio can be followed.

Annex B (normative): Binary renderer config metadata format

B.1 Definition of binary renderer config metadata format

The binary renderer config metadata format consists of acoustic environment, directivity payload components and distance attenuation components (payloadRendConfig, see Table B.1). The acoustic environment component (payloadAcEnv, see Table B.2) metadata syntax consists of a frequency grids element (payloadFreqGrid) containing single or multiple frequency grids, and a single or multiple acoustic environments. An acoustic environment contains a late reverb element (payloadLateReverb), and optionally a shoebox model element for early reflections synthesis (payloadEarlyReflections). This construction allows for dynamic switching between acoustic environments by selecting an environment using its identifier (revAcEnvID). This facilitates multiple use cases, such as scenes with multiple, fully independent rooms, dynamic scene changes, or user selectable acoustics environments. The payload syntax of the payloadAcEnv() and its elements are shown in the tables below. Locally atomic data components are marked bold with their respective size in bits and mnemonic format, and their descriptions are provided below the payload element tables. The complex payload elements are provided in subsequent tables.

Table B.1: Syntax of payloadRendConfig

Syntax	Bits	Mnemonic
payloadRendConfig() {		
if (**hasAcEnv**) {	1	bslbf
payloadAcEnv();		
}		
if (**hasDirectivity**) {	1	bslbf
payloadDirectivity();		
}		
if (**hasDistanceAttenuation**) {	1	bslbf
payloadDistanceAttenuation();		
}		
}		

Table Table B.2: Syntax of payloadAcEnv

Syntax	Bits	Mnemonic
payloadAcEnv() {		
payloadFreqGrid();		
revNrElements = GetCountOrIndex();		
for (e = 0; e < revNrElements; e++) {		
revAcEnvID[e] = GetCountOrIndex ();		
payloadLateReverb();		
if (**hasEarlyReflections**) {	1	bslbf
payloadEarlyReflections();		
}		
}		
}		

The payloadFreqGrid() element provides representation of frequency grids. There are three possible frequency grid representations possible for efficient representation: individual frequencies, start-hop-amount, and preset grid selection.

Table B.3: Syntax of payloadFreqGrid()

Syntax	Bits	Mnemonic
payloadFreqGrid () {		
fgdNrGrids = GetCountOrIndex();		
for (g = 0; g < fgdNrGrids; g++) {		
fgdMethod;	2	uimsbf
if (fgdMethod == 'Individual frequencies') {		
fgdNrBands[g] = GetCountOrIndex();		
for (b = 0; b < fgdNrBands[g]; b++) {		
fgdCenterFreq[g][b] = GetFrequency();		
}		
}		
else if (fgdMethod == 'Start-Hop-Amount') {		
fgdNrBands[g] = GetCountOrIndex();		
fgdCenterFreq[g][0] = GetFrequency();		
frequencyHop = LUT(**frequencyHopCode**);	var	vlclbf
for (b = 1; b < fgdNrBands[g]; b++) {		
fgdCenterFreq[g][b] = fgdCenterFreq[g][b - 1] * frequencyHop;		
}		
}		
else if (fgdMethod == 'Default banding') {		
fgdDefaultGrid;	4	uimsbf
if (**fgdIsSubGrid**) {	1	bslbf
fgdDefaultGridOffset;	3	uimsbf
fgdDefaultGridNrBands;	6	uimsbf
}		
}		
}		
}		

fgdMethod Indicates the method with which the frequency grid is coded.

Bits	Meaning
0b00	Individual frequencies
0b01	Start-Hop-Amount
0b10	Default banding
0b11	Reserved

LUT() Executes query on look-up table corresponding to the field whose name is provided as argument. The look-up tables used by the support elements are listed in Annex B.2.

frequencyHopCode Indicates the hop-factor for the frequency banding.

Bits	Meaning
0b0010	$2^{(1/12)}$
0b0011	$2^{(1/6)}$
0b0000	$2^{(1/4)}$
0b01	$2^{(1/3)}$
0b0001	$2^{(1/2)}$
0b11	2^{1}
0b10	2^{2}

fgdDefaultGrid Field indicating which default grid to use as frequency banding.

The preset frequency grids consist of common, perceptually relevant grids.

Table B.4: fgdDefaultGrid code table

Bits	fgdCenterFreq[g] in Hz	fgdNrBands[g]	Description
0b0000	{31.5, 63, 125, 250, 500, 1000, 2000, 4000, 8000, 16000}	10	Octave – ISO
0b0001	{25, 50, 100, 200, 400, 800, 1600, 3150, 6300, 12500}	10	Octave alternative
0b0010	{20, 25, 31.5, 40, 50, 63, 80, 100, 125, 160, 200, 250, 315, 400, 500, 630, 800, 1000, 1250, 1600, 2000, 2500, 3150, 4000, 5000, 6300, 8000, 10000, 12500, 16000, 20000}	31	1/3 octave – ISO
0b0011	{25, 100, 400, 1600, 6300}	5	2 Octave – ISO
0b0100	{125, 250, 500, 1000, 2000, 4000}	6	Octave subset
0b0101	{25, 250, 2500}	3	
0b0110	{27, 56, 89, 126, 168, 214, 265, 323, 387, 459, 539, 628, 727, 839, 963, 1101,1256, 1429, 1621, 1836, 2077, 2345, 2644, 2978, 3351, 3767, 4232, 4750, 5329, 5975, 6697, 7502, 8401, 9405, 10525, 11775, 13171, 14729, 16468, 18410, 20577}	41	1 ERB scale
0b0111	{27, 89, 168, 265, 387, 539, 727, 963, 1256, 1621, 2077, 2644, 3351, 4232, 5329, 6697, 8401, 10525, 13171, 16468, 20577}	21	2 ERB scale
0b1000	{50, 150, 250, 350, 450, 570, 700, 840, 1000, 1170, 1370, 1600, 1850, 2150, 2150, 2500, 2900, 3400, 4000, 4800, 5800, 7000, 8500, 10500, 13500}	25	Bark scale
0b1001	Reserved		
0b1010	Reserved		
0b1011	Reserved		
0b1100	Reserved		
0b1101	Reserved		
0b1110	Reserved		
0b1111	Reserved		

fgdIsSubGrid Flag indicating whether further data is present indicating a subset of the default grids.

fgdDefaultGridOffset Indicates the (0-based) index of the first relevant frequency of the default grid that is used.

fgdDefaultGridNrBands Indicates the number of bands used from the default grid.
fgdNrBands[g] = **fgdDefaultGridNrBands** + 1.

The payloadLateReverb() element contains late reverb control parameters. The RT60 and DSR parameters are provided per frequency band. These frequency bands are provided in a frequency grid as selected with revFreqGridIdx[e], which refers to an index in the frequency grid array.

Table B.5: Syntax of payloadLateReverb

Syntax	Bits	Mnemonic
payloadLateReverb() {		
revFreqGridIdx[e] = GetCountOrIndex();		
revPredelay[e] = GetDuration();		
for (b = 0; b < fgdNrBands[revFreqGridIdx[e]]; b++) {		
revRT60[e][b] = GetDuration();		
}		
for (b = 0; b < fgdNrBands[revFreqGridIdx[e]]; b++) {		
revDSR[e][b] = LUT(**dsrCode**);	var	vlclbf
}		
}		

dsrCode Code indicating the DSR value (see Table B.21).

The payloadEarlyReflections() element contains early reflections control parameters. These parameters include room dimensions, wall absorption coefficients, and optionally a listener origin. The absorption coefficients are provided per frequency band. Please note, that early reflections element uses a separate frequency grid index, since early reflections typically require lower frequency resolution than late reverb synthesis. The listener origin determines the initial listener

position in a room. Eventually, the listener position can be dynamically updated. Every acoustic environment can have an individual listener origin specified.

Table B.6: Syntax of payloadEarlyReflections

Syntax	Bits	Mnemonic
payloadEarlyReflections() {		
for (n = 0; n < 3; n++) {		
erSize[e][n] = GetDistance(true);		
}		
for (n = 0; n < 6; n++) {		
erAbsCoeff[e][n] = LUT(**absorptionCode**);	var	vlclbf
}		
If (**hasListenerOrigin**) {	1	bslfb
sign.x = **isPositiveX** ? 1 : -1;	1	bslfb
sign.y = **isPositiveY** ? 1 : -1;	1	bslfb
erListenerOrigin[e].x = sign.x * GetDistance(true);		
erListenerOrigin[e].y = sign.y * GetDistance(true);		
erListenerOrigin[e].z = GetDistance(true);		
}		
lowComplexity;	1	bslfb
}		

absorptionCode Code indicating absorption coefficients (see Table B.22).

hasListenerOrigin Indicates whether listener origin is provided. If yes, listener origin x and y coordinates are provided relative to the room center and z coordinate indicates height above the floor level.

isPositiveX, **isPositiveY** Flags indicating listener origin (x, y dimensions) offset sign.

lowComplexity Flag activating low complexity mode that favors efficient early reflection rendering over spatial accuracy.

The GetCountOrIndex() element is a basis element for providing an integer value.

Table B.7: Syntax of GetCountOrIndex

Syntax	Bits	Mnemonic
number = GetCountOrIndex() {		
number = LUT(**countOrIndexLoCode**);	var	vlclbf
if (**isLargerNumber**) {	1	bslbf
numberHi = LUT(**countOrIndexHiCode**);	var	vlclbf
number = number + numberHi * 64;		
}		
return number;		
}		

countOrIndexLoCode Code indicating the lower bits of a count or index value (see Table B.10).

isLargerNumber Flag indicating whether more bits are sent to indicate a larger number.

countOrIndexHiCode Code indicating the higher bits of a count or index value (see Table B.11).

The GetDuration() element provides means of time duration representation. Tenths of seconds are used as a leading unit as they provide the most efficient representation.

Table B.8: Syntax of GetDuration

Syntax	Bits	Mnemonic
duration = GetDuration() {		
deciSeconds = LUT(**deciSecondsCode**);	var	vlclbf
duration = deciSeconds;		
if (**addMilliseconds**) {	1	bslbf
miliSeconds = LUT(**milliSecondsCode**);	var	vlclbf
duration = duration + miliSeconds;		
if (**addMicroseconds**) {	1	bslbf
microseconds = LUT(**microsecondsCode**);	var	vlclbf
duration = duration + microseconds;		
}		
}		
if (**addSeconds**) {		
seconds = LUT(**secondsCode**);	var	vlclbf
duration = duration + seconds;		
}		
return duration;		
}		

deciSecondsCode Code for indicating decimal seconds duration offset (see Table B.12).

addMilliseconds Flag indicating whether milliseconds duration offset is transmitted next.

milliSecondsCode Code for indicating milliseconds duration offset (see Table B.13).

addMicroseconds Flag indicating whether microseconds duration offset is transmitted next.

microsecondsCode Code for indicating number of microseconds duration offset (see Table B.14).

addSeconds Flag indicating whether seconds duration offset is transmitted next.

secondsCode Code for indicating seconds duration offset (see Table B.15).

The GetDistance() element provides means of length representation. Please note, that for unconstrained acoustic environments, the isSmallScene flag shall be set to false.

Table B.8: Syntax of GetDistance

Syntax	Bits	Mnemonic
distance = GetDistance(isSmallScene) {		
meters = LUT(**metersCode**);	var	vlclbf
distance = meters;		
if (isSmallScene == false) {		
if (**addHectometers**) {	1	bslbf
hectometers = LUT(**hectometersCode**);	var	vlclbf
distance = distance + hectometers * 100;		
while (**addKilometers**) {	1	bslbf
kilometers = LUT(**kilometersCode**);	var	vlclbf
distance = distance + kilometers * 1000;		
}		
}		
}		
if (**addCentimeters**) {	1	bslbf
centimeters = LUT(**centimetersCode**);	var	vlclbf
distance = distance + centimeters / 100;		
return distance;		
}		

metersCode Code that indicates a distance in meters (see Table B.16).

addHectometers Flag that indicates whether hectometers data is available for longer distance values.

hectometersCode Code that indicates a distance in hectometers (see Table B.17).

addKilometers Flag that indicates whether kilometers data is available for very long distances.

kilometersCode Code that indicates a distance in kilometers (see Table B.18).

centimetersCode Code that indicates a distance in centimeters (see Table B.19).

The GetFrequency() elements provides means of frequency representation. The basic frequency look-up-table provides coarse one-third octave representation, whereas in case moreAccuracy flag is set to true, frequency can be refined further.

Table B.9: Syntax of GetFrequency

Syntax	Bits	Mnemonic
frequency = GetFrequency() {		
frequency = LUT(**frequencyCode**);	var	vlclbf
if (**moreAccuracy**) {	1	bslbf
frequency = frequency * 2^((**frequencyRefine** + 1) / 51);	4	uimsbf
return frequency;		
}		

frequencyCode Code that indicates a center frequency in Hz of a one-third octave band (see Table B.20)

moreAccuracy Flag that indicates whether data for a more accurate frequency is transmitted.

frequencyRefine Field that indicates a value for refining the frequency value.

The payloadDirectivity() elements describe the source directivity pattern. Each pattern has an ID and the objects can be assigned to use a specific ID.

Table B.9a1: Syntax of payloadDirectivity

Syntax	Bits	Mnemonic

payloadDirectivity() {
  directivityCount = GetCountOrIndex ();
  for (i = 0; i < directivityCount; i++) {
  directivityIndex = GetCountOrIndex ();
  ia[directivityIndex] = getAngle();
  oa[directivityIndex] = getAngle();
  og[directivityIndex] = getOuterGain();
  }
}
	
	var
	
	var
		
		vlclbf
		
		vlclbf

Table B.9a2: Syntax of GetAngle

Syntax	Bits	Mnemonic

angle = GetAngle() {
  angle = **angleCode** * 20.0;
  return angle;
}
	5	bslbf

Table B.9a3: Syntax of GetOuterGain

Syntax	Bits	Mnemonic

outerGain = GetAngle() {
  log_gain = -90.0 + **outerGainCode** * 3.0;
  outerGain = 10^(log_gain/20);
  return outerGain;
}
	5	bslbf

The payloadDistanceAttenuation describes the distance attenuation parameters that is used for all objects being rendered.

Table B.9a4: Syntax of payloadDistanceAttenuation

Syntax	Bits	Mnemonic

payloadDistanceAttenuation() {
  rd = GetRefDistMeters();
  md = GetMaxDistMeters();
  rf = GetRolloffFactor();
  }
}		

Table B.9a5: Syntax of GetRefDistMeters

Syntax	Bits	Mnemonic

refDist = GetRefDistMeters() {
  refDist = (**refDistCode** + 1) * 0.1;
  return refDist;
  }
	6	bslbf

Table B.9a6: Syntax of GetMaxDistMeters

Syntax	Bits	Mnemonic

maxDist = GetMaxDistMeters() {
  maxDist = (**maxDistCode** + 1) * 1.0;
  return maxDist;
  }
	6	bslbf

Table B.9a7: Syntax of GetRolloffFactor

Syntax	Bits	Mnemonic

rolloffFactor = GetRolloffFactor() {
rolloffFactor = **rolloffFactorCode*** 0.1;
return rolloffFactor;
}
	6	bslbf

B.2 Support Elements Look-up Tables

This clause contains the look-up tables used in the binary renderer config metadata.

Table B.10: countOrIndexLoCode look-up table

Code	Value	Code	Value	Code	Value	Code	Value	Code	Value
0111	0	001010	13	111101	26	1101000	39	1011011	52
100	1	001001	14	111100	27	1100111	40	1011010	53
01100	2	001000	15	111011	28	1100110	41	1011001	54
01101	3	000111	16	111010	29	1100101	42	1011000	55
01010	4	000110	17	111001	30	1100100	43	1010111	56
01011	5	000101	18	111000	31	1100011	44	1010110	57
01000	6	000100	19	1101111	32	1100010	45	1010101	58
01001	7	000011	20	1101110	33	1100001	46	1010100	59
001111	8	000010	21	1101101	34	1100000	47	1010011	60
001110	9	000001	22	1101100	35	1011111	48	1010010	61
001101	10	000000	23	1101011	36	1011110	49	1010001	62
001100	11	111111	24	1101010	37	1011101	50	1010000	63
001011	12	111110	25	1101001	38	1011100	51		

Table B.11: countOrIndexHiCode look-up table

Code	Value
001	1
000	2
110	3
101	4
100	5
0111	6
0101	7
1111	8
1110	9
01101	10
01001	11
01000	12
011001	13
0110001	14
0110000	15

Table B.12: deciSecondsCode look-up table

Code	Value
110	0
100	0.1
101	0.2
0110	0.3
0111	0.4
111	0.5
0100	0.6
0101	0.7
0010	0.8
0011	0.9

000	1

Table B.13: millisecondsCode look-up table

Code	Value	Code	Value	Code	Value	Code	Value	Code	Value
1111010	0	10010	0.02	10000	0.04	10110	0.06	10100	0.08
1111011	0.001	0101001	0.021	0111111	0.041	0001101	0.061	0010011	0.081
1111000	0.002	0101110	0.022	0111100	0.042	0000010	0.062	0010000	0.082
1111001	0.003	0101111	0.023	0111101	0.043	0000011	0.063	0010001	0.083
1111110	0.004	0101100	0.024	0110010	0.044	0000000	0.064	0010110	0.084
1111111	0.005	0101101	0.025	0110011	0.045	0000001	0.065	0010111	0.085
1111100	0.006	0100010	0.026	0110000	0.046	0000110	0.066	0010100	0.086
1111101	0.007	0100011	0.027	0110001	0.047	0000111	0.067	0010101	0.087
1110010	0.008	0100000	0.028	0110110	0.048	0000100	0.068	1101010	0.088
1110011	0.009	0100001	0.029	0110111	0.049	0000101	0.069	1101011	0.089
11001	0.01	10011	0.03	10001	0.05	10111	0.07	10101	0.09
1110000	0.011	0100110	0.031	0110100	0.051	0011010	0.071	1101000	0.091
1110001	0.012	0100111	0.032	0110101	0.052	0011011	0.072	1101001	0.092
1110110	0.013	0100100	0.033	0001010	0.053	0011000	0.073	1101110	0.093
1110111	0.014	0100101	0.034	0001011	0.054	0011001	0.074	1101111	0.094
1110100	0.015	0111010	0.035	0001000	0.055	0011110	0.075	1101100	0.095
1110101	0.016	0111011	0.036	0001001	0.056	0011111	0.076	1101101	0.096
0101010	0.017	0111000	0.037	0001110	0.057	0011100	0.077	1100010	0.097
0101011	0.018	0111001	0.038	0001111	0.058	0011101	0.078	1100011	0.098
0101000	0.019	0111110	0.039	0001100	0.059	0010010	0.079	110000	0.099

Table B.14: microsecondsCode look-up table

Code	Value	Code	Value	Code	Value	Code	Value	Code	Value
110111100	0.00001	110110110	0.00021	110011100	0.00041	110101010	0.00061	110100000	0.00081
10010	0.00002	001000	0.00022	000110	0.00042	011100	0.00062	111010	0.00082
110111101	0.00003	110110111	0.00023	110011101	0.00043	110101011	0.00063	110100001	0.00083
10011	0.00004	001001	0.00024	000111	0.00044	011101	0.00064	111011	0.00084
1101111110	0.00005	110110100	0.00025	110010010	0.00045	110101000	0.00065	110100110	0.00085
10000	0.00006	001110	0.00026	000100	0.00046	010010	0.00066	111000	0.00086
1101111111	0.00007	110110101	0.00027	110010011	0.00047	110101001	0.00067	110100111	0.00087
10001	0.00008	001111	0.00028	000101	0.00048	010011	0.00068	111001	0.00088
1101111100	0.00009	110011010	0.00029	110010000	0.00049	110101110	0.00069	110100100	0.00089
10110	0.00010	001100	0.00030	011010	0.00050	010000	0.00070	111110	0.00090
1101111101	0.00011	110011011	0.00031	110010001	0.00051	110101111	0.00071	110100101	0.00091
10111	0.00012	001101	0.00032	011011	0.00052	010001	0.00072	111111	0.00092
110110010	0.00013	110011000	0.00033	110010110	0.00053	110101100	0.00073	110111010	0.00093
10100	0.00014	000010	0.00034	011000	0.00054	010110	0.00074	111100	0.00094
110110011	0.00015	110011001	0.00035	110010111	0.00055	110101101	0.00075	110111011	0.00095
10101	0.00016	000011	0.00036	011001	0.00056	010111	0.00076	111101	0.00096
110110000	0.00017	110011110	0.00037	110010100	0.00057	110100010	0.00077	110111000	0.00097
001010	0.00018	000000	0.00038	011110	0.00058	010100	0.00078	11000	0.00098
110110001	0.00019	110011111	0.00039	110010101	0.00059	110100011	0.00079	110111001	0.00099
001011	0.00020	000001	0.00040	011111	0.00060	010101	0.00080		

Table B.15: secondsCode look-up table

Code	Value	Code	Value	Code	Value	Code	Value	Code	Value
0011	1	1011	7	01011	13	10101	19	101001	25
0001	2	1001	8	01001	14	011111	20	0101001	26
0000	3	1000	9	01000	15	011110	21	0101000	27
1111	4	01110	10	00101	16	010101	22	1010001	28
1101	5	01101	11	11101	17	001001	23	10100001	29
1100	6	01100	12	11100	18	001000	24	10100000	30

Table B.16: metersCode look-up table

Code	Value	Code	Value	Code	Value	Code	Value	Code	Value
111101	0	000001	20	010101	40	10011110	60	11100010	80
110010	1	000110	21	101010	41	10011111	61	11100011	81
110011	2	000111	22	101011	42	10011100	62	11100000	82
110000	3	000100	23	101000	43	10011101	63	11100001	83
110001	4	000101	24	101001	44	10010010	64	11100110	84
110110	5	011010	25	101110	45	10010011	65	11100111	85
110111	6	011011	26	101111	46	10010000	66	11100100	86
110100	7	011000	27	101100	47	10010001	67	11100101	87
110101	8	011001	28	101101	48	10010110	68	11111010	88
001010	9	011110	29	10000	49	10010111	69	11111011	89
001011	10	011111	30	1000100	50	10010100	70	11111000	90
001000	11	011100	31	1000101	51	10010101	71	11111001	91
001001	12	011101	32	10001110	52	11101010	72	11111110	92
001110	13	010010	33	10001111	53	11101011	73	11111111	93
001111	14	010011	34	10001100	54	11101000	74	11111100	94
001100	15	010000	35	10001101	55	11101001	75	11111101	95
001101	16	010001	36	10011010	56	11101110	76	11110010	96
000010	17	010110	37	10011011	57	11101111	77	11110011	97
000011	18	010111	38	10011000	58	11101100	78	11110000	98
000000	19	010100	39	10011001	59	11101101	79	11110001	99

Table B.17: hectometersCode look-up table

Code	Value
000	0
001	1
110	2
111	3
100	4
101	5
0110	6
0111	7
0100	8
0101	9

Table B.18: kilometersCode table

Code	Value
10	1
011	2
001	3
000	4
111	5
0101	6
0100	7
1101	8
11001	9
11000	10

Table B.19: centimetersCode look-up table

Code	Value	Code	Value	Code	Value	Code	Value	Code	Value
110010	0	0100110	20	0001010	40	0011110	60	100010	80
110011	1	0100111	21	0001011	41	0011111	61	100011	81
110000	2	0100100	22	0001000	42	0011100	62	100000	82
110001	3	0100101	23	0001001	43	0011101	63	100001	83
110110	4	0111010	24	0001110	44	0010010	64	100110	84
110111	5	0111011	25	0001111	45	0010011	65	100111	85
110100	6	0111000	26	0001100	46	0010000	66	100100	86
110101	7	0111001	27	0001101	47	0010001	67	100101	87
0101010	8	0111110	28	0000010	48	0010110	68	1111010	88
0101011	9	0111111	29	0000011	49	0010111	69	1111011	89
0101000	10	0111100	30	0000000	50	0010100	70	1111000	90
0101001	11	0111101	31	0000001	51	0010101	71	1111001	91
0101110	12	0110010	32	0000110	52	101010	72	1111110	92
0101111	13	0110011	33	0000111	53	101011	73	1111111	93
0101100	14	0110000	34	0000100	54	101000	74	1111100	94
0101101	15	0110001	35	0000101	55	101001	75	1111101	95
0100010	16	0110110	36	0011010	56	101110	76	111010	96
0100011	17	0110111	37	0011011	57	101111	77	111011	97
0100000	18	0110100	38	0011000	58	101100	78	111000	98
0100001	19	0110101	39	0011001	59	101101	79	111001	99

Table B.20: frequencyCode look-up table

Code	Value	Code	Value	Code	Value	Code	Value	Code	Value
100011	16	011010	80	011111	400	1101	2000	010100	10000
001110	20	011011	100	1111	500	010000	2500	010101	12500
001111	25	0001	125	011100	630	010001	3150	0010	16000
1001	31.5	011000	160	011101	800	1010	4000	10000	20000
001100	40	011001	200	1100	1000	010110	5000	10001010	25000
001101	50	1110	250	010010	1250	010111	6300	10001011	31500
0000	63	011110	315	010011	1600	1011	8000	1000100	40000

Table B.21: dsrCode look-up table

Code	Value	Code	Value	Code	Value	Code	Value	Code	Value
10001100	-150	011111010	-120	011000100	-90	110100	-60	010110	-30
10001101	-149	011111011	-119	011000101	-89	110101	-59	011010010	-29
100011110	-148	011111000	-118	011011010	-88	001010	-58	011010011	-28
100011111	-147	011111001	-117	011011011	-87	001011	-57	011010000	-27
100011100	-146	011111110	-116	011011000	-86	001000	-56	011010001	-26
100011101	-145	011111111	-115	011011001	-85	001001	-55	011010110	-25
10000010	-144	011111100	-114	011011110	-84	001110	-54	011010111	-24
10000011	-143	011111101	-113	011011111	-83	001111	-53	011010100	-23
10000000	-142	011110010	-112	011011100	-82	001100	-52	011010101	-22
10000001	-141	011110011	-111	011011101	-81	001101	-51	010111010	-21
10000110	-140	011110000	-110	010100	-80	000010	-50	010111011	-20
10000111	-139	011110001	-109	010101	-79	000011	-49	010111000	-19
10000100	-138	011110110	-108	100110	-78	000000	-48	010111001	-18
10000101	-137	011110111	-107	100111	-77	000001	-47	010111110	-17
011101010	-136	011110100	-106	100100	-76	000110	-46	010111111	-16
011101011	-135	011110101	-105	100101	-75	000111	-45	010111100	-15
011101000	-134	011001010	-104	111010	-74	000100	-44	010111101	-14
011101001	-133	011001011	-103	111011	-73	000101	-43	10001010	-13
011101110	-132	011001000	-102	111000	-72	101010	-42	10001011	-12
011101111	-131	011001001	-101	111001	-71	101011	-41	10001000	-11
011101100	-130	011001110	-100	111110	-70	101000	-40	10001001	-10
011101101	-129	011001111	-99	111111	-69	101001	-39		
011100010	-128	011001100	-98	111100	-68	101110	-38		
011100011	-127	011001101	-97	111101	-67	101111	-37		
011100000	-126	011000010	-96	110010	-66	101100	-36		
011100001	-125	011000011	-95	110011	-65	101101	-35		
011100110	-124	011000000	-94	110000	-64	010010	-34		
011100111	-123	011000001	-93	110001	-63	010011	-33		
011100100	-122	011000110	-92	110110	-62	010000	-32		
011100101	-121	011000111	-91	110111	-61	010001	-31		

Table B.22: absorptionCode look-up table

Code	Value
110	0
100	0.1
101	0.2
0110	0.3
0111	0.4
111	0.5
0100	0.6
0101	0.7
0010	0.8
0011	0.9
000	1

Table B.23: angleCode look-up table

Code	Value	Code	Value	Code	Value	Code	Value
00000	0	00101	5	01010	10	01111	15
00001	1	00110	6	01011	11	10000	16
00010	2	00111	7	01100	12	10001	17
00011	3	01000	8	01101	13	10010	18
00100	4	01001	9	01110	14		

Table B.24: outerGainCode look-up table

Code	Value	Code	Value	Code	Value	Code	Value	Code	Value	Code	Value
00000	0	00110	6	01100	12	10010	18	11000	24	11110	30
00001	1	00111	7	01101	13	10011	19	11001	25		
00010	2	01000	8	01110	14	10100	20	11010	26		
00011	3	01001	9	01111	15	10101	21	11011	27		
00100	4	01010	10	10000	16	10110	22	11100	28		
00101	5	01011	11	10001	17	10111	23	11101	29		

Table B.25: refDistCode look-up table

Code	Value	Code	Value	Code	Value	Code	Value	Code	Value	Code	Value
000000	0	001100	12	011000	24	100100	36	110000	48	111100	60
000001	1	001101	13	011001	25	100101	37	110001	49	111101	61
000010	2	001110	14	011010	26	100110	38	110010	50	111110	62
000011	3	001111	15	011011	27	100111	39	110011	51	111111	63
000100	4	010000	16	011100	28	101000	40	110100	52		
000101	5	010001	17	011101	29	101001	41	110101	53		
000110	6	010010	18	011110	30	101010	42	110110	54		
000111	7	010011	19	011111	31	101011	43	110111	55		
001000	8	010100	20	100000	32	101100	44	111000	56		
001001	9	010101	21	100001	33	101101	45	111001	57		
001010	10	010110	22	100010	34	101110	46	111010	58		
001011	11	010111	23	100011	35	101111	47	111011	59		

Table B.26: maxDistCode look-up table

Code	Value	Code	Value	Code	Value	Code	Value	Code	Value	Code	Value
000000	0	001100	12	011000	24	100100	36	110000	48	111100	60
000001	1	001101	13	011001	25	100101	37	110001	49	111101	61
000010	2	001110	14	011010	26	100110	38	110010	50	111110	62
000011	3	001111	15	011011	27	100111	39	110011	51	111111	63
000100	4	010000	16	011100	28	101000	40	110100	52		
000101	5	010001	17	011101	29	101001	41	110101	53		
000110	6	010010	18	011110	30	101010	42	110110	54		
000111	7	010011	19	011111	31	101011	43	110111	55		
001000	8	010100	20	100000	32	101100	44	111000	56		
001001	9	010101	21	100001	33	101101	45	111001	57		
001010	10	010110	22	100010	34	101110	46	111010	58		
001011	11	010111	23	100011	35	101111	47	111011	59		

Table B.27: rolloffFactorCode look-up table

Code	Value	Code	Value	Code	Value	Code	Value
000000	0	001100	12	011000	24	100100	36
000001	1	001101	13	011001	25	100101	37
000010	2	001110	14	011010	26	100110	38
000011	3	001111	15	011011	27	100111	39
000100	4	010000	16	011100	28	101000	40
000101	5	010001	17	011101	29		
000110	6	010010	18	011110	30		
000111	7	010011	19	011111	31		
001000	8	010100	20	100000	32		
001001	9	010101	21	100001	33		
001010	10	010110	22	100010	34		
001011	11	010111	23	100011	35		

Annex C (informative): Change history

Change history							
Date	**Meeting**	**TDoc**	**CR**	**Rev**	**Cat**	**Subject/Comment**	**New version**
08-2023	SA4#124 Telco	SA4a230091				Presented to Audio SWG for information	0.0.1
08-2023	SA4#125	S4-231246				Presented to SA4#125 as part of IVAS codec selection deliverables	0.1.0
08-2023	SA4#125	S4-231439				Presented to SA4#125 for agreement and for presentation to SA#101 for approval	0.2.0
09-2023	SA#101	SP-230980				Version 1.0.0 created by MCC to be sent to TSG for approval	1.0.0
09-2023	SA#101					Version 18.0.0 created by MCC upon approval	18.0.0
2024-06	SA#104	SP-240693	0002	3	B	Adding ISAR track-a split rendering feature to TS 26.258 and Corrections to the IVAS C-Code and corresponding specification text	18.1.0
2024-06	SA#104					Change of spec title as approved by TSG SA in SP-240917	18.1.0
2024-09	SA#105	SP-241115	003	1	F	Corrections to TS 26.258	18.2.0
2025-10	-	-	-	-	-	Update to Rel-19 version (MCC)	**19.0.0**
2026-01	SA#110	SP-251431	0005	2	A	Corrections to the IVAS Codec Software (floating-point), Rel. 19	**19.1.0**
2026-01	SA#110	SP-251435	0006	1	B	Correcting references for fixed-point specification	**19.1.0**
2026-03	SA#111	SP-260118	0008	1	A	Missing description of TD renderer file format	**19.2.0**

History

Version	Date	Status
V19.0.0	November 2025	Publication
V19.1.0	February 2026	Publication
V19.2.0	April 2026	Publication