INTERNATIONAL STANDARD

ISO/IEC 14496-26

First edition
2010-05-01

Information technology — Coding of audio-visual objects —

Part 26: Audio conformance

Technologies de l'information — Codage des objets audiovisuels —

Partie 26: Conformité audio

Reference number
ISO/IEC 14496-26:2010(E)

PDF disclaimer

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy. The ISO Central Secretariat accepts no liability in this area.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to the file; the PDF-creation parameters were optimized for printing. Every care has been taken to ensure that the file is suitable for use by ISO member bodies. In the unlikely event that a problem relating to it is found, please inform the Central Secretariat at the address given below.

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents

Page

Foreword v
Introduction vii
1 Scope 1
2 Normative references 1
3 Terms and definitions 2
4 Conformance Points 2
5 Profiles 4
6 Conformance data 4
6.1 File name conventions 4
6.2 Content 6
7 Audio Object Types 7
7.1 General 7
7.2 Null 14
7.3 AAC-based scalable configurations 14
7.4 AAC (main, LC, ER LC, SSR, LTP, ER LTP, ER LD, scalable, ER scalable) 15
7.5 TwinVQ and ER_TwinVQ 40
7.6 ER BSAC 44
7.7 CELP 52
7.8 ER CELP 56
7.9 HVXC 61
7.10 ER HVXC 71
7.11 ER HILN and ER Parametric 74
7.12 TTSI 89
7.13 General MIDI 91
7.14 Wavetable Synthesis 92
7.15 Algorithmic Synthesis and AudioFX 93
7.16 Main Synthetic 100
7.17 SBR 102
7.18 PS (Parametric Stereo) 113
7.19 SSC (SinuSoidal Coding) 115
7.20 DST (Lossless coding of oversampled audio) 121
7.21 Layer-3 123
7.22 ALS (Audio lossless coding) 125
7.23 SLS (Scalable Lossless Coding) 127
7.24 Layer-1 and Layer 2 130
7.25 Low Delay SBR 131
8 Audio EP tool 134
8.1 Compressed data 134
8.2 Decoders 137
9 Audio Composition 142
9.1 AudioBIFS v1 142
9.2 Advanced Audio BIFS nodes 153
9.3 AudioBIFS v3 Nodes 179
10 MPEG-4 audio transport stream 197
10.1 General 197
10.2 Compressed Data 198
10.3 Decoders 198

11 Upstream **199**
11.1 Compressed data **199**
11.2 Decoders **200**

12 Conformance test sequence assignment to profiles and levels **200**
12.1 Overview **200**
12.2 Audio **200**
12.3 Systems **216**

Annex A (informative) **Complexity measurement criteria and tool for level definitions of algorithmic synthesis and AudioFX Object Type** **221**

Annex B (informative) **Test bitstreams for the CELP object type** **242**

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

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

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

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 14496-26 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

This part of ISO/IEC 14496 cancels and replaces:

— ISO/IEC 14496-4:2004, Clause 6,

— ISO/IEC 14496-4:2004/Cor.5,

— ISO/IEC 14496-4:2004/Cor.6,

— ISO/IEC 14496-4:2004/Amd.8:2005, including ISO/IEC 14496:2004/Amd.8:2005/Cor.1:2008,

— ISO/IEC 14496-4:2004/Amd.11:2006, including ISO/IEC 14496-4:2004/Amd.11:2006/Cor.1:2008,

— ISO/IEC 14496-4:2004/Amd.11:2006/Cor.2:2007,

— ISO/IEC 14496-4:2004/Amd.11:2006/Cor.3:2008,

— ISO/IEC 14496:2004-4/Amd.13:2007, including ISO/IEC 14496-4:2004/Amd.13:2007/Cor.1:2007,

— ISO/IEC 14496:2004-4/Amd.13:2007/Cor.2:2007,

— ISO/IEC 14496-4:2004/Amd.14:2007,

— ISO/IEC 14496-4:2004/Amd.15:2007,

— ISO/IEC 14496-4:2004/Amd.18:2007,

— ISO/IEC 14496-4:2004/Amd.19:2007, including ISO/IEC 14496-4:2004/Amd.19:2007/Cor.1:2008,

— ISO/IEC 14496-4:2004/Amd.20:2008, and

— ISO/IEC 14496-4:2004/Amd.22:2008.

ISO/IEC 14496 consists of the following parts, under the general title *Information technology — Coding of audio-visual objects:*

— *Part 1: Systems*

— *Part 2: Visual*

— *Part 3: Audio*

— *Part 4: Conformance testing*

— *Part 5: Reference software*

— *Part 6: Delivery Multimedia Integration Framework (DMIF)*

— *Part 7: Optimised reference software for coding of audio-visual objects*

— *Part 8: Carriage of ISO/IEC 14496 contents over IP networks*

— *Part 9: Reference hardware description*

— *Part 10: Advanced Video Coding*

— *Part 11: Scene description and application engine*

— *Part 12: ISO base media file format*

— *Part 13: Intellectual Property Management and Protection (IPMP) extensions*

— *Part 14: MP4 file format*

— *Part 15: Advanced Video Coding (AVC) file format*

— *Part 16: Animation Framework eXtension (AFX)*

— *Part 17: Streaming text format*

— *Part 18: Font compression and streaming*

— *Part 19: Synthesized texture stream*

— *Part 20: Lightweight Application Scene Representation (LASeR) and Simple Aggregation Format (SAF)*

— *Part 21: MPEG-J Graphics Framework eXtensions (GFX)*

— *Part 22: Open Font Format*

— *Part 23: Symbolic Music Representation*

— *Part 24: Audio and systems interaction* [Technical Report]

— *Part 25: 3D Graphics Compression Model*

— *Part 26: Audio conformance*

— *Part 27: 3D Graphics conformance*

Introduction

ISO/IEC 14496-3 specifies coded representations of audio information. ISO/IEC 14496-3 allows for large flexibility, achieving suitability of ISO/IEC 14496 for many different applications. The flexibility is obtained by including parameters in the bitstream that define the characteristics of coded bitstreams. Examples are the audio sampling frequency bitrate parameters, synchronisation timestamps, the association of bitstreams and synthetic objects within objects.

This part of ISO/IEC 14496 specifies how tests can be designed to verify whether bitstreams and decoders meet the requirements as specified in ISO/IEC 14496-3 and allow interoperability with remote terminals in interactive, broadcast and local (with stored contents) sessions. These tests can be used for various purposes such as

— manufacturers of encoders, and their customers, can use the tests to verify whether the encoder produces bitstreams compliant with ISO/IEC 14496-3,

— manufacturers of decoders and their customers can use the tests to verify whether the decoder meets the requirements specified in ISO/IEC 14496-3 for the claimed decoder capabilities,

— manufacturers and customers of terminals supporting interactive, broadcast and local sessions over a multitude of transport protocols and networks, can use the tests to verify whether the claimed functionalities are compliant with ISO/IEC 14496-6,

— manufacturers of test equipments, and their customers can use the tests to verify compliance with ISO/IEC 14496-3.

Information technology — Coding of audio-visual objects —

Part 26:
Audio conformance

1 Scope

This part of ISO/IEC 14496 specifies how tests can be designed to verify whether compressed data and decoders meet requirements specified by ISO/IEC 14496-3. In this part of ISO/IEC 14496, encoders are not addressed specifically. An encoder may be said to be an ISO/IEC 14496 encoder if it generates compressed data compliant with the syntactic and semantic bitstream payload requirements specified in ISO/IEC 14496-3.

Characteristics of compressed data and decoders are defined for ISO/IEC 14496-3. The compressed data characteristics define the subset of the standard that is exploited in the compressed data. Examples are the applied values or range of the sampling rate and bitrate parameters. Decoder characteristics define the properties and capabilities of the applied decoding process. An example of a property is the applied arithmetic accuracy. The capabilities of a decoder specify which compressed data the decoder can decode and reconstruct, by defining the subset of the standard that may be exploited in the decodable compressed data. Compressed data can be decoded by a decoder if the characteristics of the compressed data are within the subset of the standard specified by the decoder capabilities.

Procedures are described for testing conformance of compressed data and decoders to the requirements defined in ISO/IEC 14496-3. Given the set of characteristics claimed, the requirements that must be met are fully determined by ISO/IEC 14496-3. This part of ISO/IEC 14496 summarises the requirements, cross references them to characteristics, and defines how conformance with them can be tested. Guidelines are given on constructing tests to verify decoder conformance. Some examples of compressed data implemented according to these guidelines are provided as an electronic annex to this document usually together with their uncompressed counterparts (reference waveforms).

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 11172-3, *Information technology — Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s — Part 3: Audio*

ISO/IEC 11172-4, *Information technology — Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s — Part 4: Compliance testing*

ISO/IEC 13818-3, *Information technology — Generic coding of moving pictures and associated audio information — Part 3: Audio*

ISO/IEC 13818-4, *Information technology — Generic coding of moving pictures and associated audio information — Part 4: Conformance testing*

ISO/IEC 13818-7, *Information technology — Generic coding of moving pictures and associated audio information — Part 7: Advanced Audio Coding (AAC)*

ISO/IEC 14496-1, *Information technology — Coding of audio-visual objects — Part 1: Systems*

ISO/IEC 14496-3, *Information technology — Coding of audio-visual objects — Part 3: Audio*

ISO/IEC 14496-11, *Information technology — Coding of audio-visual objects — Part 11: Scene description and application engine*

3 Terms and definitions

For the purposes of this document the terms, defintions, symbols and abbreviated terms given in ISO/IEC 14496-1, ISO/IEC 14496-3 and the following apply.

3.1
conformance data
conformance test sequences and **conformance tools**

3.2
conformance tool
tool to check certain conformance criteria

NOTE Conformance tools are provided in the electronic attachments to this part of ISO/IEC 14496.

3.3
conformance test sequence
superset of **compressed data** and its **reference waveforms**

NOTE Examples of conformance test sequences are provided in the electronic attachments to this part of ISO/IEC 14496.

3.4
compressed data
data encoded in accordance with ISO/IEC 14496-3

3.5
reference waveform
decoded counterparts of the **compressed data**

4 Conformance Points

All audio decoders except the LATM-based decoders are part of the MPEG-4 framework. Table 1 gives an overview about the interfaces that have to be provided to test the audio decoders using the MPEG-4 System.

Table 1 — Conformance points

conformance point/interface	data flow direction	description/reference
AudioSpecificConfig	in	audio related decoder specific information, see ISO/IEC 14496-3:2009, (1.6.2.1 AudioSpecificConfig)
audio access units	in	audio related bitstream payload, see ISO/IEC 14496-1:2004 (7.1.2.3 Access Units (AU))
BIFS/AudioSource node	in	see ISO/IEC 14496-11: 2005 (7.2.2.15 Audio Source)
private test info	in	to control some elements which are usually generated by random number generators
audio composition units	out	see ISO/IEC 14496-1: 2004 (7.2.8 Composition Units (CU))

Figure 1 gives an overview about the test bench (MPEG-4 System), the system under test (Audio decoder), and the interfaces between them. Figure 2 gives a more detailed view on the audio decoder, consisting of error protection (EP) decoder and audio core decoder.

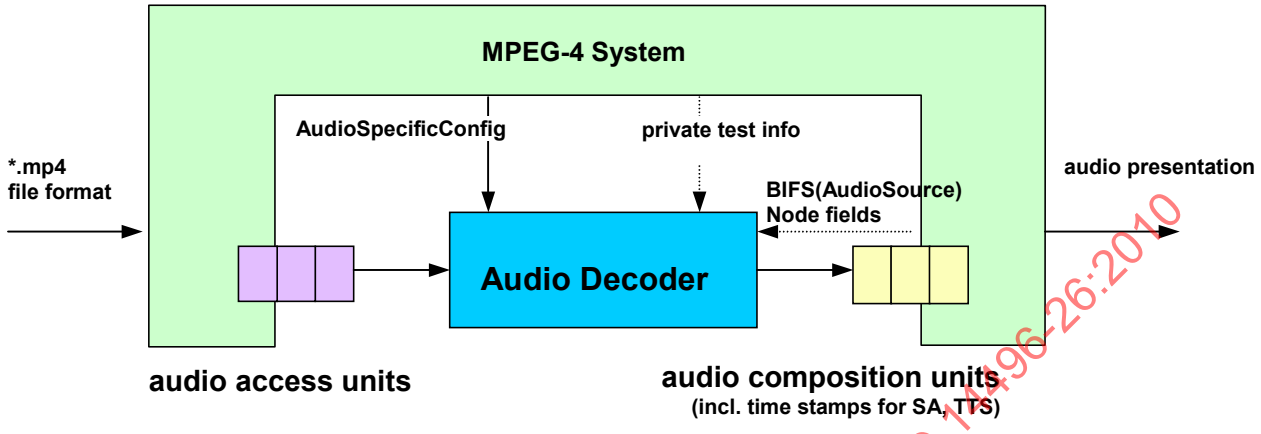

Figure 1 — Audio Conformance Points

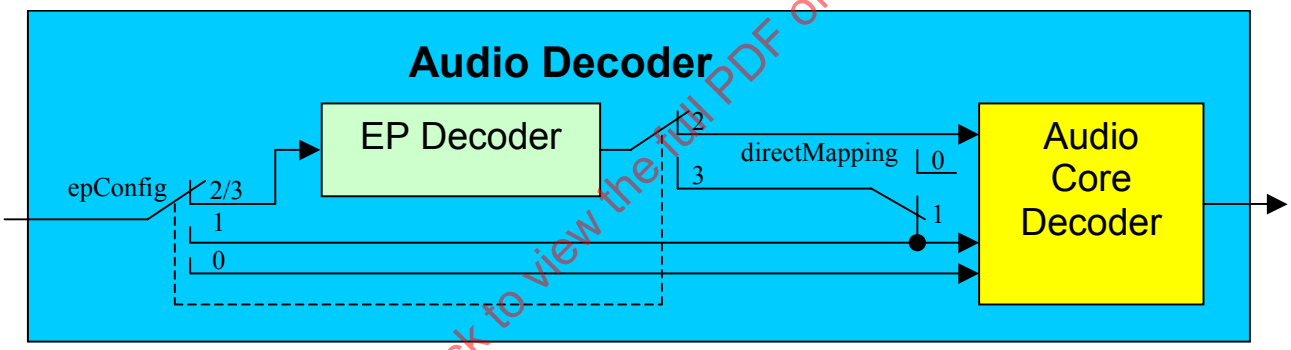

Figure 2 — Audio decoder structure

Clause 7 describes:

The conformance criteria of the audio core decoder.

The conformance criteria of the compressed data not requiring the EP decoder (epConfig == 0 || epConfig == 1).

The properties of the examples of compressed data with (epConfig == 0 || epConfig == 1).

Clause 8 describes:

The conformance criteria of the EP decoder

The conformance criteria of the compressed data requiring the EP decoder (epConfig == 2 || epConfig == 3).

The properties of the examples of compressed data with (epConfig == 2 || epConfig == 3).

Compressed data with different epConfig settings might be available referring to the same reference waveforms. Here, the output of a conforming decoder shall be equal, independently of the used epConfig setting.

For some of the compressed data containing scalable configurations, conformance points are defined at the PCM output of the decoder for *m* layers being decoded from an *n*-layer input, where *m* is an integer in the range 0 (base layer conformance) to *n-1*. The reference PCM decoder output signals corresponding to these conformance points are listed in the respective conformance tables.

5 Profiles

ISO/IEC 14496-3 defines several profiles and several levels within each profile. Conformance is always tested against a certain level within a certain profile. Audio profiles always comprise a set of audio object types. Nevertheless the conformance criteria as described within this document are based on audio object types. The assignment of object types to profiles as well as the level definitions can be found in ISO/IEC 14496-3. The conformance of a certain level within a certain profile is fulfilled, if the conformance of each object type belonging to this profile is fulfilled. The assignment of the provided test sequences to profiles and levels can be found in Clause 12.

6 Conformance data

6.1 File name conventions

For all conformance test sequences, the file name convention given in Table 2 is used.

Table 2 — File name conventions

object type name/ tool name	File Name (compressed)	File Name (uncompressed)
AdvancedAudioBIFS - perceptual apporach	aabper<coreSetup>	-- not applicable --
AdvancedAudioBIFS - physical approach	aabphy<coreSetup>	-- not applicable --
AudioBIFS	ab<coreSetup>_<coder>	ab<coreSetup>_<coder>
AudioBIFS v3	ABv3_<nodeAbbrev><coreSetup>	-- not applicable --
AAC scalable	ac<coreSetup>	ac<coreSetup>[_lay<highestLay>]
AAC LC	al<coreSetup>_<fs>	al<coreSetup>_<fs>[_cut<fac>_boost<facr>] [_level<lvl>][_<chan>]
AAC main	am<coreSetup>_<fs>	am<coreSetup>_<fs>[_cut<fac>_boost<facr>] [_level<lvl>][_<chan>]
AAC LTP	ap<coreSetup>_<fs>	ap<coreSetup>_<fs>
AAC SSR	as<coreSetup>_<fs>	as<coreSetup>_<fs>[_<chan>]
CELP	ce<coreSetup>	ce<coreSetup>[_lay<highestLay>]
ER AAC scalable	er_ac<coreSetup>_ep<epConfig> [<epSetup>]	er_ac<coreSetup>[_lay<highestLay>]
ER AAC LD	er_ad<coreSetup>_<fs>_ep<epConfig> [<epSetup>]	er_ad<coreSetup>_<fs>
ER AAC LC	er_al<coreSetup>_<fs>_ep<epConfig> [<epSetup>]	er_al<coreSetup>_<fs>
ER AAC LTP	er_ap<coreSetup>_<fs>_ep<epConfig> [<epSetup>]	er_ap<coreSetup>_<fs>
SBR (+AAC LC)	al_sbr_<tool>_<fs>_<nchan>[_fsaac<fs>][_sig<sig>]	al_sbr_<mode>_<tool>_<fs>_<nchan> [_fsaac<fs>][_sig<sig>][_<chan>]
SBR (+AAC LC with 960 samples per frame)	al960_sbr_<tool>_<fs>_<nchan> [_fsaac<fs>][_sig<sig>]	al960_sbr_<mode>_<tool>_<fs>_<nchan> [_fsaac<fs>][_sig<sig>][_<chan>]
PS (+SBR+AAC LC)	al_sbr_ps_<coreSetup>	al_sbr_ps_<coreSetup>[_<version>]
SSC	ssc_<tool>_<nchan>[_sig<sig>]	ssc_<mode>_<tool>_<nchan>[_sig<sig>][_<chan>]
DST	dst_<tool>_<nchan>[_sig<sig>]	dst_<mode>_<tool>_<nchan>[_sig<sig>][_<chan>]
Layer-3	l3_<coreSetup>	l3_<coreSetup>

ER BSAC	er_bs<coreSetup>_<fs>_ep<epConfig> [<epSetup>]	er_bs<coreSetup>_<fs>[_lay<highestLay>]
ER CELP	er_ce<coreSetup>_ep<epConfig> [<epSetup>]	er_ce<coreSetup>[_lay<highestLay>]
ER HILN	er_hi<coreSetup>_ep<epConfig> [<epSetup>]	er_hi<coreSetup>[_lay<highestLay>] [_s<speedFac>][_p<pitchFac]
ER HVXC	er_hv<coreSetup>_ep<epConfig> [<epSetup>]	er_hv<coreSetup>[_lay<highestLay>]_<delay>
ER Parametric	er_pa<coreSetup>_ep<epConfig> [<epSetup>]	er_pa<coreSetup>[_lay<highestLay>]_<delay>
ER Twin VQ	er_tv<coreSetup>_ep<epConfig> [<epSetup>]	er_tv<coreSetup>[_lay<highestLay>]
HVXC	hv<coreSetup>	hv<coreSetup>[_lay<highestLay>]_ref<decCfg>
Algorithmic Synthesis and Audio FX	sy<coreSetup>	sy<coreSetup>
TTSI	tts<coreSetup>	tts<coreSetup>
TwinVQ	tv<coreSetup>	tv<coreSetup>[_lay<highestLay>]
ALS	als_<tool>_<coreSetup>	als_<tool>_<coreSetup>
SLS	sls<coreSetup>_<fs>_<bitres>	sls<coreSetup>_<fs>_<bitres>
Layer-1	l1_<coreSetup>	l1_<coreSetup>
Layer-2	l2_<coreSetup>	l2_<coreSetup>
ER AAC ELD	er_eld<coreSetup>_<fs>_ep<epConfig> [<epSetup>]	er_eld<coreSetup>_<fs>

<bitres> can be 16 or 24 and indicates the bit resolution of the coded wavefile

<chan> indicates the channel for multi-channel sequences (f<number> - number of the front channel, b<number>- number of the back channel, s<number> - number of the side channel, l<number> - number of the LSF channel).

<coder> indicates the coder used to encode the content (ce – CELP, sa – Structured Audio, pcm – PCM)

<coreSetup> refers to a certain audio coder setup. It is most likely a number, but might also contain characters.

<delay> refers to the decoder delay, it can become “ld” (low delay) or “nd” (normal delay).

<epConfig> can be 0, 1, 2 or 3, depending on epConfig (defined in AudioSpecificConfig).

<epSetup> is required if (epConfig==2 || epConfig==3). It refers to a certain error protection setup.

<fs> sampling frequency (08, 11, 12, 16, 22, 24, 32, 44, 48, 64, 88 or 96).

_level<lvl> refers to the level with regard to DRC.

_cut<fac>_boost<fac> referes to the cut and boost factors with regard to DRC.

_lay<highestLay> is required for any scalable configuration. It marks the highest layer of the scalable configuration used for decoding (starting with 0 for the core layer).

_p<pitchfac> is a number refering to the decoder configuration with regard to the pitch factor.

_ref<decCfg> is a number refering to the decoder configuration with regard to delay mode, speed and pitch change.

_s<speedfac> is a number refering to the decoder configuration with regard to the speed factor.

<tool> indicates the SBR module mainly targeted by the test sequence. Possible values are “e” for testing the envelope adjuster “s” for testing sine addition, “gh” for testing time-grid transitions in combination with changes of SBR header data, “i” for testing inverse filtering, “qmf” for testing the QMF implementation, “cm” for testing various channel modes, “sig” for testing SBR signaling, “twi” for QMF identification, and “sr” for testing various combinations of sampling rates.

<nodeAbbrev> is the abbreviation of one of the AudioBIFS v3 node names.

<nchan> corresponds to the number of channels present in the conformance test sequence. It is either a single integer, in which case it refers to the number of main audio channels, or two integers separated by a “.”, in which case the first integer equals the number of main audio channels, while the second number equals the number of low frequency enhancement channels.

fsaac<fs> corresponds to the sampling rate of the underlying AAC-LC data. If it is omitted, it is half the sampling rate given as output sampling rate.

<sig> is an integer describing the kind of signalling used according to the table below. If this value is omitted, backwards compatible explicit signalling of SBR is used.

file name conventions

sig	Signalling method used
0	Implicit signalling of SBR
1	Hierarchical explicit signalling of SBR
2	Backwards compatible explicit signalling of SBR

<mode> is either “hq” or “lp” for the high quality or the low power version of the SBR decoding algorithm respectively.

<version> is either “bl” or “ur” for the baseline or the unrestricted version of the parametric stereo decoding algorithm respectively.

With respect to file extensions, the following rules are applied:

Compressed	MPEG-4 file format	.mp4
Compressed	native MPEG-1/2 Audio storage format	.mpg
Compressed	Audio data interchange format	.adif
Compressed	Audio data transport stream	.adts
Compressed	AudioSyncStream	.ass
Compressed	EPAudioSyncStream	.ess
Compressed	AudioPointerStream	.aps
Uncompressed	HILN Conformance Test Parameters	.ctp
Uncompressed	WAVE format (uncompressed PCM format)	.wav
Uncompressed	TTSI decoded text and control digits	.txt

6.2 Content

The test set includes a set of sine sweeps, a set of musical/speech test sequences and a set of noise-like test sequences. The supplied sine sweeps with an amplitude of -20dB relative to full scale have an absolute amplitude of +/- 0.1.

7 Audio Object Types

7.1 General

This Clause lists all audio object types. It starts with a general description, which may be related to more than one object type.

This Clause contains general descriptions for conformance testing on compressed data and decoders. Unless explicitly restricted, these descriptions are related to all object types.

7.1.1 Compressed Data

7.1.1.1 Characteristics

Characteristics of compressed data specify the constraints that are applied by the encoder in generating the compressed data. These syntactic and semantic constraints may, for example, restrict the range or the values of parameters that are encoded directly or indirectly in the compressed data. The constraints applied to a given compressed data may or may not be known a priori.

Decoder relevant compressed data may consist of the following parts:

decoder specific information (AudioSpecificConfig)

BIFS/AudioSource node (field information)

audio access units (establishing the bitstream payload)

7.1.1.1.1 ESC instance configuration

In case of epConfig=1, each instance of each sensitivity category belonging to one frame is stored separately within a single access unit, i.e. there exist as many elementary streams as instances defined within a frame.

Note: In case of epConfig=3, the mapping between EP classes and ESC instances is signaled by the data element directMapping. In case of directMapping=1, the restrictions regarding the ESC instance configuration apply accordingly to the EP class configuration.

The following table gives an overview about the valid configurations:

Table 3 — Number of ESC instances that build a frame in case of epConfig==1

Audio object type	number of ESC instances to build a frame
ER AAC	see Table 4
ER Twin VQ	non-scalable or base layer: 2 any enhancement layer: 2
ER BSAC	base layer: 2 any large-step enhancement layer: 1
ER CELP	base layer: 5 any enhancement layer: 1
ER HVXC	2 kbit/s, non-scalable or base layer: 4 4 kbit/s, non-scalable: 5 any enhancement layer: 3
ER HILN	base layer: 5 any enhancement/extension layer: 1
ER Parametric	PARAmode==0,1 base layer: 5 PARAmode==2,3 base layer: 15 any enhancement/extension layer: 1

Table 4 — Number of ESC instances that build elements/layers of an ER AAC frame in the case of epConfig==1

	aacScalefactorDataResilienceFlag	
	0	1
single channel element (SCE) / mono layer	3	4
channel pair element (CPE) / stereo layer	7	9
extension payload (EPL)	2	

Depending on the value of the data element channelConfiguration, an AAC frame might cover several instances of SCE, CPE or EPL. This leads to the following valid configurations:

Table 5 — Number of ESC instances that build an ER AAC frame/layer in the case of epConfig==1

AOT					aacScalefactorDataResilienceFlag		
					0	1	
17	19	20	23	channelConfiguration	main payload		N extension payloads
x	x	x	x	1	3	4	
x	x	x	x	2	7	9	
x	x		x	3	3+7	4+9	
x	x		x	4	3+7+3	4+9+4	+2*N
x	x		x	5	3+7+7	4+9+9	
x	x		x	6	3+7+7+3	4+9+9+4	
x	x		x	7	3+7+7+7+3	4+9+9+9+4	

7.1.1.2 Test procedure

Each compressed data shall meet the syntactic and semantic requirements specified in ISO/IEC 14496-3. For each audio object type a set of semantic tests to be performed on the compressed data is described. To verify whether the syntax is correct is straightforward and therefore not defined herein after. In the description of the semantic tests it is assumed that the tested compressed data contains no errors due to transmission or other causes. For each test the condition or conditions that must be satisfied are given, as well as the prerequisites or conditions in which the test can be applied.

7.1.2 Decoders

7.1.2.1 Characteristics

The decoder characteristics are defined by the profiles and levels being tested.

7.1.2.2 Test procedure

To test audio decoders, ISO/IEC JTC 1/SC 29/WG 11 supplies a number of test sequences. Supplied sequences cover all profile decoders. For a supplied test sequence, testing can be done by comparing the output of a decoder under test with a reference output also supplied by ISO/IEC JTC 1/SC 29/WG 11. In cases where the decoder under test is followed by additional operations (e.g. quantizing a signal to a 16 bit output signal) the conformance point is prior to such additional operations, i.e. it is permitted to use the actual decoder output (e.g. with more than 16 bit) for conformance testing.

Measurements are carried out relative to full scale where the output signals of the decoders are normalized to be in the range between –1.0 and +1.0.

The following Subclauses define a set of test methods. A particular test method for a certain test sequence is specified in the object type specific subclauses.

For elements producing output that cannot be tested with the methods described below, specific conformance testing procedures are described in the object type specific subclauses.

7.1.2.2.1 RMS/LSB Measurement

To fulfill the “RMS/LSB Measurement” test at an accuracy level of “K bit”, an ISO/IEC 14496-3 decoder shall provide an output waveform such that the RMS level of the difference signal between the output of the decoder under test and the supplied reference output is less than $2^{-(K-1)}/\mathrm{sqrt}(12)$. In addition, the difference signal shall have a maximum absolute value of at most $2^{-(K-2)}$ relative to full-scale. The “RMS/LSB Measurement” test shall be carried out for an accuracy level of K=16 bit unless a different accuracy level is explicitly stated.

7.1.2.2.1.1 Calculation of RMS

For the calculation of the RMS level, all measurements are carried out relative to full scale where the output signals of the decoder and supplied test sequences are normalized to be in the range between -1.0 and +1.0.

The supplied reference waveforms have a precision (P) of 24 bits, where the most significant bit (MSB) will be labeled bit 0 and the least-significant bit (LSB) will be labeled bit 23. The most significant bit (bit 0) represents the value of –1, the second most significant bit (bit 1) represents the value of +1/2, etc.

$$\begin{aligned}
\text{value of bit 0 (MSB)} &= -\frac{1}{2^0} = -1 \\
\text{value of bit 1} &= \frac{1}{2^1} = \frac{1}{2} \\
\text{value of bit 2} &= \frac{1}{2^2} = \frac{1}{4} \\
&\vdots \\
\text{value of bit 23 (LSB)} &= \frac{1}{2^{23}} = \frac{1}{8{,}388{,}608}
\end{aligned}$$

The output waveform of the decoder under test is required to be in the same format. In the case that the output of the decoder has a precision of P' bits and if P' is smaller than 24, then the output is extended to 24 bits by setting bit P' through bit 23 to zero. In the next step, the difference (*diff*) of the samples of these signals has to be calculated. Every channel of a multichannel waveform shall be tested. The total number of samples for each channel is N.

$$diff(n) = \text{' output signal of decoder under test } (n)\text{'} - \text{' supplied test sequence } (n)\text{'}, \text{ for } n = 1 \text{ to } N$$

The values of all difference samples shall be squared, summed, divided by N and then the square-root shall be calculated. This calculation finally gives the RMS level.

$$rms = \sqrt{\frac{1}{N}\sum_{n=1}^{N} diff(n)^2}$$

This test only verifies the computational accuracy of an implementation.

Software is provided for performing this verification procedure.

7.1.2.2.2 Segmental SNR

This criterion is designed to test decoders decoding the object types CELP, ER CELP, HVXC, ER HVXC, TwinVQ, ER TwinVQ and ER HILN.

Definition:

$x_a(i)$: i^{th} sample of reference output signal (normalized in a range between –1.0 and 1.0).

$x_b(i)$: i^{th} sample of output signal of a decoder under test normalized in a range between –1.0 and 1.0.

L : the length of segment

N : the total number of segments

$SS(k)$: SNR of k^{th} segment

$SSNR$: segmental SNR

$$SS(k) = \log_{10}\left(1 + \frac{\sum_{i=0}^{L-1} x_a(k \times L + i)^2}{10^{-13} L + \sum_{i=0}^{L-1} \left(x_a(k \times L + i) - x_b(k \times L + i)\right)^2}\right)$$

$$SSNR = 10 \times \log_{10}\left(10^{\sum_{k=0}^{N-1} SS(k)/N} - 1.0\right)$$

7.1.2.2.3 Frequency domain criterion based on cepstrum analysis

This criterion is designed to test decoders decoding the object types CELP, ER CELP, TwinVQ, ER TwinVQ and ER HILN.

The cepstrum analysis procedure is defined by means of the functions lpc2cepstrum and calculate_lpc provided in pseude C code below.

```
#define LPC_ORDER                       16   /*    LPC order                 */
#define CEPSTRUM_ORDER                  32   /*    Cepstrum order            */
#define BW                              0.0125F  /*    Bandwidth scalefactor      */


void lpc2cepstrum (float    lpc_coef[],     /*    in:    LPC coefficients (a-parameters)    */
                   float    C[])            /*    out:   LPC cepstrum                       */

{
    float ss;
    int   i, m;

    /* it is assumed that lpc_coef[0] is 1 ! */

    C[1] = -lpc_coef[1];

    for (m = 2; m <= LPC_ORDER; m++)
    {
        ss= -lpc_coef[m] * m;
        for (i = 1; i < m; i++)
        {
            ss -= lpc_coef[i] * C[m-i];
        }
        C[m] = ss;
    }

    for (m = LPC_ORDER + 1; m <= CEPSTRUM_ORDER; m++)
    {
        ss = 0.0F;
        for (i = 1; i<= LPC_ORDER; i++)
        {
            ss -= lpc_coef[i] * C[m-i];
        }
        C[m] = ss;
    }

    for (m = 2; m <= CEPSTRUM_ORDER; m++)
    {
        C[m] /= m;
    }
}


void calculate_lpc (float    *in,           /*    in:    input PCM audio data               */
                    int      frame_size,    /*    in:    analysis frame length in samples   */
                    float    *lpc_coef)     /*    out:   LPC coefficients                   */

{
    int      ip;
    float    wvpowfr, cor[LPC_ORDER + 1];
    float    wlag [LPC_ORDER + 1];
    float    *wdw;

    wdw = (float*) malloc (sizeof (float) * frame_size);

    if (wdw == NULL)
    {
        printf ("Memory allocation error in calculate_lpc.\n");
        exit (1);
    }

    hamwdw (wdw, frame_size);

    for (ip = 0; ip < frame_size; ip++)
    {
        in[ip] *= wdw[ip];
    }

    sigcor (in, frame_size, &wvpowfr, cor, LPC_ORDER);

    lagwdw (wlag, LPC_ORDER, BW);
```

```
    for (ip = 1; ip <= LPC_ORDER; ip++)
    {
        cor[ip] *= wlag[ip];
    }

    corref (LPC_ORDER, cor, lpc_coef);

    free (wdw);
}

void hamwdw (float    wdw[],
             int      n)
{
    int        i;
    float      d, pi = 3.141592653589793F;

    d = (float) (2.0 * pi/n);

    for (i = 0; i < n; i++)
    {
        wdw[i] = (float) (0.54 - 0.46 * cos (d * i));
    }
}
void lagwdw (float    wdw[],
             int      n,
             float    h)
{
    int      i;
    float    pi = 3.141592653589793F;
    float    a, b, w;

    a = (float) (log (0.5) * 0.5 / log (cos (0.5 * pi * h)));
    a = (float) ((int) a);
    w = 1.0F;
    b = a;
    wdw[0] = 1.0F;

    for (i = 1; i <= n; i++)
    {
        b += 1.0F;
        w *= a / b;
        wdw[i] = w;
        a -= 1.0F;
    }
}

void sigcor (float    *sig,
             int      n,
             float    *_pow,
             float    cor[],
             int      p)
{
    int      k, ij;
    float    c, dsqsum;
    float    sqsum = 1.0e-35F;

    if (n > 0)
    {
        for (ij = 0; ij < n; ij++)
        {
            sqsum += (sig[ij] * sig[ij]);
        }
        dsqsum = (float) (1.0 / sqsum);

        for (k = 1; k <= p; k++)
        {
            c = 0.0;
            for(ij = k; ij < n; ij++)
            {
                c += (sig[ij - k] * sig[ij]);
            }
            cor[k] = c * dsqsum;
        }
        k = p;
    }
    *_pow = (float) ((sqsum - 1.e-35) / (float)n);
```

```
    cor[0] = 1.0F;

}

void corref (int      p,            /*    in:    LPC analysis order              */
             float    cor[],        /*    in:    correlation coefficients        */
             float    alf[])        /*    out:   linear predictive coefficients  */

{
    int      i, j, k;
    float    resid, r, a;
    float    ref[LPC_ORDER + 1];

    ref[1] = cor[1];
    alf[1] = -ref[1];
    resid  = (float) ((1.0 - ref[1]) * (1.0 + ref[1]));

    for (i = 2; i <= p; i++)
    {
        r = cor[i];

        for (j = 1; j < i; j++)
        {
            r += alf[j] * cor[i-j];
        }

        alf[i] = -(ref[i] = (r /= resid ));
        j = 0;
        k = i;

        while (++j <= --k)
        {
            a = alf[j];
            alf[j] -= r * alf[k];

            if (j < k)
            {
                alf[k] -= r * a;
            }
        }
        resid = (float) (resid * (1.0 - r) * (1.0 + r));
    }
}
```

7.1.2.2.4 PNS conformance criteria

Two tests based on spectral waveform analysis and one test based on temporal waveform analysis shall be applied.

Spectral PNS conformance analysis:

[PNS-1] Both the decoded output and the reference output signal are analyzed by means of an N-point DFT (N=2*number_of_spectral_lines_per_frame, e. g. 2048-point for AAC LC) with a Hann window and 50 % overlap between subsequent windows. For both signals, the DFT lines are grouped corresponding to scalefactor bands and the accumulated squared absolute values are computed for each scalefactor band. As the first test criterion, the ratio between the energies of both signals averaged over time shall be within the interval [-0.4 dB; 0.4 dB] for each scalefactor band. As the second test criterion, the ratio between the standard deviations (over time) of the energies of both signals shall be within the interval [-0.8 dB; 0.8 dB] for each scalefactor band. For this test, sequences are supplied containing a static spectrum generated by a single PNS codebook section covering all scalefactor bands (i.e. each frame carries the same spectral “envelope”, long blocks only, no other codebooks).

[PNS-2] The same type of analysis and the same thresholds are used as in test [PNS-1], but with a window size of N/8 and grouping corresponding to scalefactor bands for a SHORT_WINDOW. For this test, sequences are supplied containing a periodic repetition of PNS and Null codebook sections within grouped short blocks ({1;1;1;1;2;2} grouping with PNS switched on in subblocks 0,2,4).

Temporal PNS conformance analysis:

[PNS-3] Starting at the first available decoder frame boundary, the sum of the squared output samples is computed for blocks of number_of_spectral_lines_per_frame/16 samples for both decoded signal and reference signal. As the test criterion, the ratio between the energies of both signals shall be within the interval [-5 dB;5 dB] for 91 % of the blocks and within the interval [-10 dB;10 dB] for 99 % of the blocks. For this test, the same sequences as provided for [PNS-2] shall be used.

7.2 Null

The NULL object type provides the possibility to feed raw PCM data directly into the audio compositor. No decoding is involved. The sampling rate and the audio channel configuration is specified by the AudioSpecificConfig.

7.3 AAC-based scalable configurations

7.3.1 Compressed data

7.3.1.1 Characteristics

Encoders may apply restrictions to the following parameters of the compressed data.

7.3.1.1.1 Layer configuration

number of non-AAC layers

number of AAC layers

7.3.1.1.2 AudioSpecificConfig

See description for individual object types.

7.3.1.1.3 Bitstream payload

See description for individual object types.

7.3.1.2 Test procedure

Each compressed data shall meet the syntactic and semantic requirements specified in ISO/IEC 14496-3. This Subclause describes a set of semantic tests to be performed on decoder relevant data. The procedure to verify whether the syntax is correct is straightforward and therefore not defined in this Subclause. In the description of the semantic tests it is assumed that the tested compressed data contains no errors due to transmission or other causes. For each test the condition or conditions that must be satisfied are given, as well as the prerequisites or conditions in which the test can be applied.

7.3.1.2.1 Layer configuration

The number of AAC layers shall not exceed 8 in any scalable configuration.

The number of CELP layers (object type 8 or 24) shall not exceed 2, if CELP is used as base layer coder within an AAC based scalable configuration.

The number of TwinVQ layers (object type 7 or 21) shall not exceed 1, if TwinVQ is used as base layer coder within an AAC based scalable configuration.

Only those object type combinations shown in Table 6 are valid.

Table 6 — Valid object type combinations within an AAC-based scalable configuration

audio object type for the base layer coder	audio object type for the AAC enhancement layers
6 (AAC scalable)	6 (AAC scalable)
8 (CELP)	6 (AAC scalable)
7 (TwinVQ)	6 (AAC scalable)
20 (ER AAC scalable)	20 (ER AAC scalable)
24 (ER CELP)	20 (ER AAC scalable)
21 (ER TwinVQ)	20 (ER AAC scalable)

If CELP is used as base layer coder within an AAC based scalable configuration, its samplingFrequencyIndex shall be 0xc (7350 Hz) or 0xb (8000 Hz).

7.3.1.2.2 AudioSpecificConfig

7.3.1.2.2.1 AudioSpecificConfig()

channelConfiguration: Shall be 1 in case of audioObjectType 7 (TwinVQ) or 21 (ER TwinVQ).

7.3.1.2.3 Bitstream payload

7.3.1.2.3.1 tvq_scalable_main_header()

tns_data_present: Shall be 0.

7.3.1.2.3.2 aac_scalable_main_header()

max_sfb: Shall not be smaller than last_max_sfb (helper variable specified in ISO/IEC 14496-3).

7.3.1.2.3.3 aac_scalable_extension_header()

max_sfb: Shall not be smaller than last_max_sfb (helper variable specified in ISO/IEC 14496-3).

7.4 AAC (main, LC, ER LC, SSR, LTP, ER LTP, ER LD, scalable, ER scalable)

7.4.1 Compressed data

7.4.1.1 Characteristics

Encoders may apply restrictions to the following parameters of the compressed data:

7.4.1.1.1 AudioSpecificConfig

a) samplingFrequencyIndex
b) samplingFrequency
c) channelConfiguration

d) program_config_element()
e) frameLengthFlag
f) dependsOnCoreCoder
g) extensionFlag
h) epConfig
i) ErrorProtectionSpecificConfig()
j) aacSectionDataResilienceFlag
k) aacScalefactorDataResilienceFlag
l) aacSpectralDataResilienceFlag

7.4.1.1.2 Bitstream payload

a) use of prediction in main profile
b) pulse_data
c) window_shape
d) M/S stereo
e) intensity stereo
f) TNS
g) data_stream_element()
h) dependently switched coupling channel
i) independently switched coupling channel
j) LFE channel
k) matrix-downmix

7.4.1.2 Test procedure

Each compressed data shall meet the syntactic and semantic requirements specified in ISO/IEC 14496-3. This Subclause describes a set of semantic tests to be performed on decoder relevant data. The procedure to verify whether the syntax is correct is straightforward and therefore not defined in this Subclause. In the description of the semantic tests it is assumed that the tested compressed data contains no errors due to transmission or other causes. For each test the condition or conditions that must be satisfied are given, as well as the prerequisites or conditions in which the test can be applied.

7.4.1.2.1 AudioSpecificConfig

7.4.1.2.1.1 AudioSpecificConfig()

audioObjectType: Shall be encoded according to the AAC object type (see Table 9).

samplingFrequencyIndex: Shall be encoded with the values specified in Table 7.

samplingFrequency: Shall be encoded with the values specified in Table 7.

extensionAudioObjectType: Shall be the Audio Object Type SBR (AOT == 5).

extensionSamplingFrequency: Shall be encoded with a value listed in Table 7, and the value shall be the same as samplingFrequency, or twice the value of samplingFrequency.

extensionSamplingFrequencyIndex: Shall be encoded with a value listed in Table 7, and the value shall indicate an extensionSamplingFrequency being the same as samplingFrequency as indicated by samplingFrequencyIndex, or the value shall indicate an extensionSamplingFrequency being twice the value of samplingFrequency.

sbrPresentFlag: Shall be encoded with the value zero if no SBR data is contained in the compressed MPEG-4 data. If SBR data is present in the compressed MPEG-4 data the parameter shall be encoded with the value one.

Table 7 — Specification of samplingFrequencyIndex and samplingFrequency

SamplingFrequencyIndex / SamplingFrequency	Level 1	Level 2	Level 3	Level 4
Scalable Profile	0x6..0xc, 0xf / <= 24000		0x3..0xc, 0xf / <= 48000	
Main Profile	0x0..0xc, 0xf / no limitation			

SamplingFrequencyIndex / SamplingFrequency	Level 1,5	Level 2,6	Level 3,7	Level 4,8
High Quality Audio Profile	0x7..0xc, 0xf / <= 22050	0x3..0xc, 0xf / <= 48000		
Low Delay Audio Profile	0xb..0xc, 0xf / <= 8000	0x8..0xc, 0xf / <= 16000	0x3..0xc 0xf / <= 48000	

SamplingFrequencyIndex/ SamplingFrequency	Level 1,3	Level 2,6
Natural Audio Profile	0x3..0xc 0xf / <=48000	0x0..0xc, 0xf / <=96000

SamplingFrequencyIndex/ SamplingFrequency	Level 1,4	Level 2,5	Level 3,6
Mobile Audio Internet Working Profile	0x6..0xc, 0xf / <= 24000	0x3..0xc, 0xf / <= 48000	

SamplingFrequencyIndex / SamplingFrequency	Level 1	Level 2	Level 3	Level 4	Level 5
AAC Profile	0x6..0xc, 0xf / <= 24000	0x3..0xc, 0xf / <= 48000	NA	0x3..0xc, 0xf / <= 48000	0x0..0xc, 0xf / <= 96000

samplingFrequencyIndex / samplingFrequency		Level 1	Level 2	Level 3	Level 4	Level 5
High Efficiency AAC Profile	SBR present	NA	0x6..0xc, 0xf / <= 24000	0x3..0xc, 0xf / <= 48000	0x3..0xc, 0xf / <= 48000 (Note 1)	0x3..0xc, 0xf / <= 48000
	SBR not present	NA	0x3..0xc, 0xf / <= 48000	0x3..0xc, 0xf / <= 48000	0x3..0xc, 0xf / <= 48000	0x0..0xc, 0xf / <= 96000
Note 1: For Level 4, for one or two channels the maximum AAC sampling rate, with SBR present, is 48 kHz. For more than two channels the maximum AAC sampling rate, with SBR present, is 24 kHz. (0x6..0xc, 0xf / <= 24000)						

extensionSamplingFrequencyIndex / extensionSamplingFrequency	Level 1	Level 2	Level 3,4	Level 5
High Efficiency AAC Profile	NA	0x6..0xc, 0xf / <= 24000	0x3..0xc, 0xf / <= 48000	0x0..0xc, 0xf / <= 96000

channelConfiguration: shall be encoded with the values specified in Table 8. In the case of channelConfiguration=0, the following restrictions apply to the number of syntactic elements specified in the program_config_element():

- the number of main audio channels (represented by SCE and CPE) shall not exceed the maximum number specified for a certain profile and level.
- the number of remaining audio channels (represented by LFE and CCE) shall not exceed the maximum number specified for a certain AudioObjectType and number of main audio channels (see ISO/IEC 14496-3, subclause “Levels within the Profiles”).

Table 8 — Specification of ChannelConfiguration

ChannelConfiguration	Level 1	Level 2	Level 3	Level 4
Scalable Profile	0, 1	0..2		0..7
Main Profile	0..7			

ChannelConfiguration	Level 1, 5	Level 2, 6	Level 3, 7	Level 4, 8
High Quality Audio Profile	0,1	0..2	0..6	
Low Delay Audio Profile	0,1			0..2

ChannelConfiguration	Level 1, 2, 3, 4
Natural Audio Profile	0..7

ChannelConfiguration	Level 1, 4	Level 2, 5	Level 3, 6
Mobile Audio Internet Working Profile	0, 1	0..2	0..6

ChannelConfiguration	Level 1	Level 2	Level 3	Level 4	Level 5
AAC Profile	0..2	0..2	NA	0..6	0..6
High Efficiency AAC Profile	NA	0..2	0..2	0..6	0..6

In addition to this table, the following audioObjectType based restrictions apply:

- channelConfiguration=0 is permitted only for the audioObjectTypes 1 (AAC main), 2 (AAC LC), 3 (AAC SSR) and 4 (AAC LTP), but not for the audioObjectTypes 6 (AAC scalable), 17 (ER AAC LC), 19 (ER AAC LTP), 20 (ER AAC scalable) and 23 (ER AAC LD).
- channelConfiguration>2 is not permitted for audioObjectTypes 6 (AAC scalable) and 20 (ER AAC scalable).

epConfig: No restrictions apply.

directMapping: Shall be 1.

7.4.1.2.1.2 GASpecificConfig()

frameLengthFlag: Shall be zero for the following audio object types: 1, 2, 3, 4, 17, 19, when used in Scalable Audio Profile, Main Audio Profile, High Quality Audio Profile, Natural Audio Profile or Mobile Audio Internet Working Profile. No restrictions apply otherwise.

dependsOnCoreCoder: Shall be encoded with the value 1 in the first AAC scalable coding layer (audio object type 6 or 20) if a core coder is used in the underlying base layer of a scalable AAC configuration; shall be encoded with the value 0 otherwise.

coreCoderDelay: no restrictions apply.

extensionFlag: shall be encoded with the value 0 in the case of the audioObjectTypes 1, 2, 3, 4, 6. Shall be encoded with the value 1 in the case of the audioObjectTypes 17, 19, 20, 23.

extensionFlag3: Shall be encoded with the value 0.

7.4.1.2.1.3 ELDSpecificConfig()

frameLengthFlag: no restrictions apply.

aacSectionDataResilienceFlag: no restrictions apply

aacScalefactorDataResilienceFlag: no restrictions apply

aacSpectralDataResilienceFlag: no restrictions apply

ldSbrPresentFlag: no restrictions apply

ldSbrSamplingRate: no restrictions apply

ldSbrCrcFlag: no restrictions apply

eldExtType: no restrictions apply

eldExtLenAdd: no restrictions apply

eldExtLenAddAdd: no restrictions apply

other_byte: no restrictions apply

7.4.1.2.1.4 program_config_element()

No program may contain more main audio channels, LFE channels, independent coupling_channel_element()'s and dependent coupling_channel_element()'s than specified by the profile and level.

The following restrictions apply to the elements of program_config_element():

element_instance_tag: no restrictions

object_type: shall match the AudioObjectType within AudioSpecificConfig

sampling_frequency_index: shall match the samplingFrequencyIndex within AudioSpecificConfig

num_front_channel_elements: see restriction regarding the number of channels as stated above.

num_side_channel_elements: see restriction regarding the number of channels as stated above.

num_back_channel_elements: see restriction regarding the number of channels as stated above.

num_lfe_channel_elements: see restriction regarding the number of channels as stated above.

num_assoc_data_elements: no restrictions apply.

num_valid_cc_elements: see restriction regarding the number of channels as stated above.

mono_mixdown_present: shall be 0 for the audio object types 1 (AAC main), 2 (AAC LC), 3 (AAC SSR) and 4 (AAC LTP) when used in Scalable Audio Profile, Main Audio Profile, High Quality Audio Profile or Natural Audio Profile.

mono_mixdown_element_number: shall be encoded with the element_instance_tag of a single_channel_element().

stereo_mixdown_present: shall be 0 for the audio object types 1 (AAC main), 2 (AAC LC), 3 (AAC SSR) and 4 (AAC LTP) when used in Scalable Audio Profile, Main Audio Profile, High Quality Audio Profile or Natural Audio Profile.

stereo_mixdown_element_number: shall be encoded with the element_instance_tag of a channel_pair_element.

matrix_mixdown_idx_present: may only be encoded with a value of 1 if a 3 front/2 rear 5-channel program (with or without LFE) is indicated for this PCE.

matrix_mixdown_idx: no restrictions apply.

pseudo_surround_enable: no restrictions apply.

front_element_is_cpe[i]: no restrictions apply.

front_element_tag_select: shall be encoded with the element_instance_tag of either a single_channel_element or a channel_pair_element.

side_element_is_cpe[i]: no restrictions apply.

side_element_tag_select: shall be encoded with the element_instance_tag of either a single_channel_element or a channel_pair_element.

back_element_is_cpe[i]: no restrictions apply.

back_element_tag_select: shall be encoded with the element_instance_tag of either a single_channel_element or a channel_pair_element.

lfe_element_tag_select: shall be encoded with the element_instance_tag of a lfe_channel_element.

assoc_data_element_tag_select: shall be encoded with the element_instance_tag of a data_stream_element.

cc_element_is_ind_sw: shall be encoded with the same value as the ind_sw_cce_flag field of the coupling_channel_element corresponding to valid_cc_element_tag_select.

valid_cce_element_tag_select: shall be encoded with the element_instance_tag of a coupling_channel_element.

comment_field_bytes: no restrictions apply.

comment_field_data[i]: no restrictions apply.

7.4.1.2.1.5 ErrorProtectionSpecificConfig()

number_of_concatenated_frame: Shall be one.

For details see also Clause 8.

7.4.1.2.2 Bitstream payload

7.4.1.2.2.1 raw_data_block()

id_syn_ele: if a program_config_element() (PCE) is present, it shall be the first syntactic element in a raw_data_block(), indicated by id_syn_ele encoded with a value of ID_PCE

7.4.1.2.2.2 Any syntactic element

element_instance_tag: ensure that element_instance_tag numbers within each element type are unique within each frame. This restriction does not apply to data_stream_element()'s (DSE), which may have duplicated element_instance_tags.

7.4.1.2.2.3 channel_pair_element()

common_window: no restrictions apply.

ms_mask_present: shall not be encoded with the binary value 11.

ms_used: no restrictions apply.

7.4.1.2.2.4 ics_info()

ics_reserved_bit: shall be set to zero.

window_sequence: Shall be zero (ONLY_LONG_SEQUENCE) if audioObjectType == 23, no such restriction applies for the remaining object types. The meaningful window_sequence transitions are as follows:

from ONLY_LONG_SEQUENCE to $\begin{cases}\text{ONLY\_LONG\_SEQUENCE}\\ \text{LONG\_START\_SEQUENCE}\end{cases}$

from LONG_START_SEQUENCE to $\begin{cases}\text{EIGHT\_SHORT\_SEQUENCE}\\ \text{LONG\_STOP\_SEQUENCE}\end{cases}$

from LONG_STOP_SEQUENCE to	{ONLY_LONG_SEQUENCE LONG_START_SEQUENCE
from EIGHT_SHORT_SEQUENCE to	{EIGHT_SHORT_SEQUENCE LONG_STOP_SEQUENCE

Other, non-meaningful, window_sequence transitions are also possible:

from ONLY_LONG_SEQUENCE to	{EIGHT_SHORT_SEQUENCE LONG_STOP_SEQUENCE
from LONG_START_SEQUENCE to	{ONLY_LONG_SEQUENCE LONG_START_SEQUENCE
from LONG_STOP_SEQUENCE to	{EIGHT_SHORT_SEQUENCE LONG_STOP_SEQUENCE
from EIGHT_SHORT_SEQUENCE to	{ONLY_LONG_SEQUENCE LONG_START_SEQUENCE

A conformant bitstream shall consist of only meaningful window_sequence transitions. However, decoders are required to handle non-meaningful window_sequence transitions as well. Test sequences al03 and as17 are provided respectively for AOT 2 (AAC LC) and AOT 3 (AAC SSR) to test decoder performance on non-meaningful window sequence transitions (note that AOT 1 (AAC Main) and AOT 4 (AAC LTP) decoders also need to fulfil conformance for AOT 2) (see 7.4.1.2.2.1). The performance requirements for non-meaningful window_sequence transitions are the same as for the meaningful transitions.

window_shape: no restrictions apply.

max_sfb: shall be <= num_swb_long or num_swb_short as appropriate for window_sequence and sampling frequency.

scale_factor_grouping: no restrictions apply.

predictor_data_present: shall be encoded with the value 0 for the audioObjectTypes 2 (AAC LC), 3 (AAC SSR) and 17 (ER AAC LC); shall be encoded with the value 0 when used in the Low Delay AAC profile; no restrictions apply otherwise.

predictor_reset: shall be encoded with the binary value of 1 sufficiently often so that normative behaviour is achieved (AAC main).

predictor_reset_group_number: shall not be encoded with the binary values 00000 or 11111 (AAC main).

prediction_data_used: no restrictions apply.

ltp_data_present: No restrictions apply.

7.4.1.2.2.5 pulse_data()

number_pulse: no restrictions apply.

pulse_start_sfb: shall be smaller than num_swb_long_window[fs_index].

pulse_offset[i]: swb_offset_long_window[pulse_start_sfb] + pulse_offset[0] + ... + pulse_offset[number_pulse] shall not be greater than 1023.

pulse_amp[i]: shall be encoded with a value small enough such that the compensated quantized spectral coefficient is not greater than 8191.

7.4.1.2.2.6 coupling_channel_element()

The number of dependently-switched and independently-switched coupling channel elements shall not exceed the allowed numbers specified by the profile and level. No coupling channel shall target a given single_channel_element() or channel_pair_element() more than once per frame. Dependently switched coupling channels are not permitted for audio object type 4 (AAC LTP).

ind_sw_cce_flag: shall not be encoded with the binary value of 1 if independently-switched coupling channel elements are not specified by the level and profile.

num_coupled_elements: shall not be encoded with a value greater than the total number of single_channel_elements and channel_pair_elements.

cc_target_is_cpe: shall be encoded with the binary value 1 if the syntactic element with element_instance_tag of cc_target_tag_select is a channel_pair_element; otherwise, it shall be encoded with the binary value of 0.

cc_target_tag_select: shall only be encoded with a binary value equal to the element_instance_tag of a single_channel_element or a channel_pair_element of the current frame.

cc_l: no restrictions apply.

cc_r: no restrictions apply.

cc_domain: no restrictions apply.

gain_element_sign: no restrictions apply.

gain_element_scale: no restrictions apply.

common_gain_element_present: no restrictions apply.

hcod_sf: see 7.4.1.2.2.17.

7.4.1.2.2.7 lfe_channel_element()

The number of LFEs shall not exceed the allowed number specified by the profile & level.

The **window_shape** field of any LFE shall always be encoded with a value of 0 (sine window).

The **window_sequence** field of any LFE shall always be encoded with a value of ONLY_LONG_SEQUENCE.

Only the lowest 12 spectral coefficients of any LFE may be non-zero.

The **predictor_data_present_flag** of any LFE shall be encoded with a value of 0.

Temporal noise shaping shall not be used in any LFE.

7.4.1.2.2.8 data_stream_element()

data_byte_align_flag: no restrictions apply.

count: no restrictions apply.

esc_count: no restrictions apply.

dat_stream_byte: no restrictions apply.

7.4.1.2.2.9 fill_element()

count: no restrictions apply.

esc_count: no restrictions apply.

Fill elements containing an extension_payload with an extension_type of EXT_SBR_DATA or EXT_SBR_DATA_CRC shall not contain any other extension_payload of any other extension_type. For fill elements containing an extension_payload with an extension_type of EXT_SBR_DATA or EXT_SBR_DATA_CRC, the fill_element count field shall be set equal to the total length in bytes, including the SBR enhancement data plus the extension_type field.

7.4.1.2.2.10 gain_control_data()

For the audio object type AAC SSR the following restrictions apply:

aloccode: shall satisfy the following conditions:

$$\text{aloccode}[B][w][m_1] < \text{aloccode}[B][w][m_2],\ 1 \le m_1 < m_2 \le \text{adjust\_num}[B][w]+1$$

where B is the Band ID, an integer between 1 and 3, and w is the Window ID, an integer from 0 to 7.

No restrictions apply for the remaining data elements inside of gain_control_data().

7.4.1.2.2.11 aac_scalable_main_header()

ics_reserved_bit: see 7.4.1.2.2.4.

window_sequence: see 7.4.1.2.2.4.

window_shape: see 7.4.1.2.2.4.

max_sfb: see 7.4.1.2.2.4.

scale_factor_grouping: see 7.4.1.2.2.4.

ms_mask_present: see 7.4.1.2.2.4.

tns_channel_mono_layer: no restrictions apply.

tns_data_present: see 7.4.1.2.2.13.

ltp_data_present: Shall be zero if audioObjectType == 20 (ER AAC scalable). No restrictions apply otherwise.

7.4.1.2.2.12 aac_scalable_extension_header()

max_sfb: see 7.4.1.2.2.4.

ms_mask_present: see 7.4.1.2.2.4.

tns_data_present: see 7.4.1.2.2.13.

7.4.1.2.2.13 diff_control_data()

diff_control: no restrictions apply.

7.4.1.2.2.14 diff_control_lr()

diff_control_lr: no restrictions apply.

7.4.1.2.2.15 individual_channel_stream()

global_gain: no restrictions apply.

pulse_data_present: shall be encoded with a value of 0 for the audioObjectTypes 6 (AAC scalable) and 20 (ER AAC scalable); shall be encoded with a value of 0 when used in the Low Delay AAC profile; shall be encoded with a value of 0 if window_sequence is EIGHT_SHORT_SEQUENCE; no restrictions apply otherwise.

tns_data_present: no restrictions apply.

gain_control_data_present: no restrictions apply for AAC SSR; otherwise it shall be encoded with the value 0.

length_of_reordered_spectral_data: Shall be equal to the length of the reordered spectral data. In case of a SCE or LFE it shall be <= 6144. In case of CPE the sum of both values shall be <= 12288.

length_of_longest_codeword: Shall reflect the length of the longest codeword transmitted within the current frame. It shall be <= 48.

7.4.1.2.2.16 section_data()

sect_cb[g][i]: Shall not be encoded with the decimal value 12 (bit sequence either “1100” (aacSectionDataResilienceFlag == 0) or “01100” (aacSectionDataResilienceFlag == 1)).

Intensity codebooks INTENSITY_HCB and INTENSITY_HCB2 shall not occur in a single_channel_element, the left channel of a channel pair element, a coupling channel element, or an LFE. Intensity codebooks can only occur in a channel_pair_element if the common_window field is set to 1.

Given that ms_used[g][sfb] is set to 1 or ms_mask_present equals the binary value 10, sfb_cb[g][sfb] shall not equal NOISE_HCB in only one channel of a channel pair element.

sect_len_incr: The sum of all sect_len_incr elements for a given window group shall equal max_sfb.

7.4.1.2.2.17 scale_factor_data()

hcod_sf[ ]: Shall only be encoded with the values listed in the scalefactor Huffman table. Shall be encoded such that the decoded scalefactors sf[g][sfb] are within the range of zero to 255, both inclusive.

dpcm_noise_nrg: No restrictions apply.

sf_concealment: No restrictions apply.

rev_global_gain: Shall be encoded with the PCM value of the last scale factor.

length_of_rvlc_sf: Shall be equal to the length of the RVLC data part in bits.

rvlc_cod_sf: Shall only be encoded with the values listed in the RVLC codebook table.

sf_escapes_present: No restrictions apply.

length_of_rvlc_escapes: Shall be equal to the length of the RVLC escape data part in bits.

rvlc_esc_sf: Shall only be encoded with the values listed in the Huffman codebook table for RVLC escape values.

dpcm_is_last_position: Shall be encoded with the last intensity stereo position.

dpcm_noise_last_position: Shall be encoded with the last noise energy value.

7.4.1.2.2.18 tns_data()

n_filt: no restrictions apply.

coef_res: no restrictions apply.

length[w][filt]: shall be small enough such that the lower bound of the filtered region, indicated by 'bottom', does not exceed the start of the array containing the spectral coefficients (spec[w])

order[w][filt]: shall not exceed the maximum permitted order depending on the specified object type and sampling frequency

direction: no restrictions apply.

coef_compress: no restrictions apply.

coef: no restrictions apply.

7.4.1.2.2.19 ltp_data()

No restrictions apply to any of the data elements inside ltp_data().

7.4.1.2.2.20 spectral_data()

hcod[sect_cb[g][i]][w][x][y][z]: shall only be encoded with the values listed in Huffman codebooks 1, 2, 3, or 4.

quad_sign_bits: no restrictions apply.

hcod[sect_cb[g][i]][y][z]: shall only be encoded with the values listed in Huffman codebooks 5 through 11.

pair_sign_bits: no restrictions apply.

hcod_esc_y: shall be encoded with a value smaller or equal to 8191, i.e., it shall be encoded with an initial escape sequence consisting of not more than nine "1" bits followed by an escape separator of "0".

hcod_esc_z: shall be encoded with a value smaller or equal to 8191, i.e., it shall be encoded with an initial escape sequence consisting of not more than nine "1" bits followed by an escape separator of "0".

7.4.1.2.2.21 extension_payload()

extension_type: no restrictions apply.

fill_nibble: shall be "0000".

fill_byte: shall be "10100101".

data_element_version: shall be "0000".

dataElementLengthPart: no restrictions apply.

data_element_byte: no restrictions apply.

other_bits: no restrictions apply.

7.4.1.2.2.22 dynamic_range_info()

No restrictions apply to any of the data elements inside dynamic_range_info().

7.4.1.2.2.23 excluded_channels()

No restrictions apply to any of the data elements inside excluded_channels().

7.4.1.2.2.24 ms_data()

ms_used: see 7.4.1.2.2.3.

7.4.1.2.2.25 channel_pair_element_eld()

max_sfb: shall be <= num_swb_long as appropriate for sampling frequency.

ms_mask_present: shall not be encoded with the binary value 11

ms_used: no restrictions apply.

7.4.1.2.2.26 individual_channel_stream_eld()

global_gain: no restrictions apply

max_sfb: shall be <= num_swb_long as appropriate for sampling frequency.

tns_data_present: no restrictions apply

length_of_reordered_spectral_data: Shall be equal to the length of the reordered spectral data. In case of a SCE or LFE it shall be <= 6144. In case of CPE the sum of both values shall be <= 12288.

length_of_longest_codeword: Shall reflect the length of the longest codeword transmitted within the current frame. It shall be <= 48.

7.4.2 Decoders

7.4.2.1 Characteristics

The object types AAC LC (Low Complexity), AAC main, AAC SSR (Scalable Sampling Rate) and AAC LTP (Long Term Prediction) build the basic object types supporting AAC-based audio coding within MPEG-4 using the ISO/IEC 13818-7 style syntax. The AAC Scalable Object type is built on top of the AAC LTP object type, but uses a different decoder structure, syntax and additional tools to provide large step scalability.

The AAC LC, AAC main and AAC SSR object types correspond to the LC, Main, SSR profiles of ISO/IEC 13818-7, with the inclusion of PNS as a mandatory tool in MPEG-4 AAC decoders. The AAC main and AAC LTP object types are built on top of the AAC LC object type. The AAC SSR is identical to the AAC LC object type with the exception of the filterbank, the additional gain control tool and some aspects of the TNS tool configuration. All these object types have an ISO/IEC 13818-7 syntax style.

Table 9 — AAC Object Types

Audio Object Type	GA Bitstream Syntax Type	Hierarchy	Object Type ID
AAC main	ISO/IEC 13818-7 Style	contains AAC LC	1
AAC LC	ISO/IEC 13818-7 Style		2
AAC SSR	ISO/IEC 13818-7 Style		3
AAC LTP	ISO/IEC 13818-7 Style	contains AAC LC	4
AAC scalable	Scalable		6

Four ER AAC object types have been defined in ISO/IEC 14496-3. Table 10 gives an overview.

Table 10 — Overview about the AAC object types

Audio Object Type	Object Type ID	13818-7 LC	PNS	LTP	TLSS	Low Delay AAC	Error Robust	Low Delay SBR
ER AAC LC	17	X	X				X	
ER AAC LTP	19	X	X	X			X	
ER AAC scalable	20	X	X		X		X	
ER AAC LD	23		X	X		X	X	
ER AAC ELD	39		X			X	X	X

The object types ER AAC LC, ER AAC LTP, and ER AAC scalable are based on the object types AAC LC, AAC LTP, and AAC scalable respectively as defined in ISO/IEC 14496-3. The object type ER AAC LD is based on the object type ER AAC LTP, but introduces some changes in order to reduce the overall algorithmic delay. Table 11 shows these dependencies.

Table 11 — AAC object type dependencies

ER Audio object type	underlying Audio object type
ER AAC LC	AAC LC
ER AAC LTP	AAC LTP
ER AAC scalable	AAC scalable
ER AAC LD	ER AAC LTP

All ER AAC object types use the error resilient bitstream payload syntax. This syntax is based on the syntax of the underlying non-ER AAC object types. The error resilient bitstream payload can be derived by subdivision of the bitstream payload data elements into instances of error sensitivity classes.

The error resilient bitstream payload is mandatory.

Beside the error resilient bitstream syntax, modified noiseless coding tools are introduced for section data, scale factor data, and spectral data. These tools are optional.

In general, conformance criteria defined for the underlying object types are also valid for the Version 2 object types and will not be repeated here. Thus, characteristics defined for a new object type have to be treated as extensions or modifications with respect to the already defined characteristics of the underlying object type.

A compliant decoder may also support any of the following modifications to the parameters in an audio bitstream:

Table 12 — AAC Parameter

Bitstream Characteristic	Variation
program configuration	any configuration of compressed data containing more than one program (in the sense of what is specified in a program_config_element()) is not allowed if a program_config_element() is used, syntactic elements (other than ID_FILL or ID_END) not referenced by any program_config_element() are not allowed
data_stream_element	a decoder is not required to store or present data recovered from data_stream_element()'s
mono-mixdown element	a decoder conforming with one of the currently defined profiles is not required to support compressed data containing any mono-mixdown element
stereo-mixdown element	a decoder conforming with one of the currently defined profiles is not required to support compressed data containing any stereo-mixdown element
matrix-mixdown	a decoder is not required to calculate a matrix-mixdown signal

The channel configuration information given in a program_config_element() inside an AAC payload conveyed as an MPEG-4 access unit shall be ignored. Note that in this case, the channel configuration information is already available from the MPEG-4 decoder configuration, specifically the GASpecificConfig(), which can include a program_config_element(). The channel configuration information given in a program_config_element() inside an AAC payload is only evaluated in case of an MPEG-4 ADTS bitstream with channel_configuration == 0, since such a bitstream does not include a GASpecificConfig().

Note that, next to channel configuration information, a program_config_element() can carry additional information, e.g. coefficients for a matrix mixdown or a comment field. This additional information can be conveyed dynamically by a program_config_element() inside an MPEG-4 access unit as well as by a program_config_element() included in the GASpecificConfig(). Such information does not affect the normative behavior of a decoder and may hence be ignored. However it may be utilized for non-normative decoder operation modes like matrix mixdown.

7.4.2.1.1 AAC main

The MPEG-4 AAC main object type is the counterpart to the MPEG-2 AAC Main Profile, though also offering the PNS tool. The AAC main object type bitstream syntax is compatible with the syntax defined in ISO/IEC 13818-7. All the MPEG-2 AAC multi-channel capabilities are available. A decoder capable of decoding a MPEG-4 Main Access Unit can also parse and decode an MPEG-2 AAC Main Profile raw_data_stream(). On the other hand, an MPEG-2 Main profile decoder will not be able to parse an MPEG-4 AAC Main stream if PNS has been used. The AAC main Object Type is an extension of the AAC LC Object Type.

7.4.2.1.2 AAC LC

The MPEG-4 AAC Low Complexity (LC) object type is the counterpart to the MPEG-2 AAC Low Complexity Profile, though also offering the PNS tool. The AAC LC object type bitstream syntax is compatible with the syntax defined in ISO/IEC 13818-7. All the MPEG-2 AAC multi-channel capabilities are available. A decoder capable of decoding an MPEG-4 LC Access Unit can also parse and decode an MPEG-2 AAC LC Profile raw_data_stream(). On the other hand, an MPEG-2 AAC LC profile decoder will not be able to parse an MPEG-4 AAC-LC stream if PNS has been used.

7.4.2.1.3 AAC SSR

The MPEG-4 AAC Scalable Sampling Rate (SSR) object type is the counterpart to the MPEG-2 AAC SSR Profile, though also offering the PNS tool. The AAC SSR object type bitstream syntax is compatible with the syntax defined in ISO/IEC 13818-7. All the MPEG-2 SSR multi-channel capabilities are available. A decoder capable of decoding a MPEG-4 SSR Access Unit can also parse and decode a MPEG-2 SSR Main profile raw data stream. On the other hand, an MPEG-2 SSR profile decoder will not be able to parse an MPEG-4 AAC-SSR stream if PNS has been used.

7.4.2.1.4 AAC LTP

The AAC LTP Object Type is an extension of the AAC LC Object Type with a long term predictor. At the same time, the MPEG-4 AAC LTP object type is similar to the AAC main object type. However, an LTP replaces the MPEG-2 AAC predictor and the PNS tool can be used in addition. The LTP achieves a similar coding gain, but requires significantly lower implementation complexity. The bitstream syntax for this object type is very similar to the syntax defined in ISO/IEC 13818-7. An MPEG-2 AAC LC profile bitstream can be decoded without restrictions by an LTP decoder.

The decoder shall use the MPEG-4 long term predictor.

7.4.2.1.5 AAC scalable

The scalable AAC object type is built on top of the AAC LTP object type, but uses a different bitstream syntax, decoder structure and additional tools to support bitrate- and bandwidth- scalability. A large number of scalable combinations are available, including combinations with TwinVQ and CELP coder tools. However, only mono or 2-channel stereo objects are supported.

AAC-based scalable configurations shall support all object type combinations specified in ISO/IEC 14496-3. All AAC and TwinVQ layers and their enhancement layers need to operate at the same sampling rate. In case of using a CELP core coder the ratio between CELP core and AAC enhancement layer sampling rates is restricted according to the specification in the General Audio part of ISO/IEC 14496-3.

7.4.2.2 Test procedure

The test procedures specified in Table 15 has to be applied. The RMS test procedure always includes the LSB test. Table 13 provides the references to the according test specifications.

Table 13 — References to the test precedure descriptions

Name of the test procedure as used in Table 15	Reference to the test specification
RMS	7.1.2.2.1
PNS	7.1.2.2.4

If no test is specified, a check of conformance using appropriate measurements, e.g. the LSB criterion (for those sequences that do not utilize PNS) or objective perceptual measurement systems, is not mandatory but highly recommended. This also applies to bitstreams with non-meaningful window sequences.

7.4.2.3 Test sequences

To test AAC decoders, ISO/IEC JTC 1/SC 29/WG 11 supplies a number of test sequences. The test sequences are defined in Table 14 and Table 15. In the case that the ChannelConfiguration equals zero the program_config_element() is defined in Table 16. Sequences are provided at sampling rates of 8, 11.025, 12, 16, 22.05, 24, 32, 44.1, 48, 64, 88.2, and 96 kHz for the audio object types AAC main, AAC LC, AAC SSR, AAC scalable, ER AAC LC, ER AAC LTP and ER AAC scalable and at sampling rates of 22.05, 24, 32, 44.1 and 48 kHz for the audio object type ER AAC LD and ER AAC ELD. The extension _fs is appended to the sequence name to indicate the sampling rate of the test sequence. Possible values of fs are 08, 11, 12, 16, 22,

24, 32, 44, 48, 64, 88 and 96, corresponding to the possibly non-integer sampling rates listed above. If two bitrates are listed in the table for a certain sequence, the lower bitrate is to be used for sampling rates of 16 kHz and below, and the higher bitrate is to be used at sampling rates above 16 kHz.

All sequences for the object types AAC main, AAC LC, AAC SSR, AAC scalable, ER AAC LC, ER AAC LTP and ER AAC LD and ER AAC ELD are supplied in all sampling rates (as indicated by extension _fs). For a specific profile and level only the sequences with the appropriate channel configuration and sampling rate are applicable.

Some conformance test sequences have special properties as follows:

Dynamic Range Control: This field indicates that dynamic range control information is available in the bitstream payload of the compressed data. Conformant decoders must be able to parse these test sequences. However, DRC semantics is optional, making the result of decoding the DRC test sequences informative only.

Arithmetic torture: This field indicates that as many different Huffman codewords from the spectrum Huffman codebooks as possible are used within the bitstream payload of the compressed data. At least 95 % of the total number of the individual codewords is processed.

Buffer test: This field indicates that the bitstream payload of the compressed data is intended to check the decoder's input buffer size. The number of remaining bits in the bit reservoir are first kept at a high level and than at one or several blocks within the stream all accumulated free bits are used by a single raw_data_block(). Thus, the resulting block lengths for these raw_data_block()'s are similar to the total decoder input buffer size, or, as this might be difficult to achieve, not more than 16 bit below that value.

Non-meaningful window_sequence transitions: This field indicates that the bitstream payload of the compressed data is intended to check the decoder's behavior in case of a non-meaningful window sequence transitions (see 7.4.1.2.2.1 for details).

Table 14 a) — AAC test sequences

file base name	content	bitrate (kbit/s)	AudioObjectType	SamplingFrequencyIndex	ChannelConfiguration	frameLengthFlag	coreCoderDelay	intensity	MS	window sequence switching	non-meaningful window_sequence transitions	window shape switching	tns_data_present	pulse data	prediction	LTP	PNS	data stream elements	gain compensation enabled	dynamic range control	bandwidth	buffer test	arithmetic torture	test procedure
am00	sine sweep	40/64	1	*	0	0	-	-	-	y	n	y	n	n	y	n	n	n	-	n	-	n	n	RMS
am01	music	40/64	1	*	0	0	-	-	-	y	n	y	n	n	y	n	n	n	-	n	-	n	n	none
am02	music	80/128	1	0..3,6..11	0	0	-	y	y	y	n	y	y	n	y	n	n	n	-	n	-	n	n	none
am02	music	128	1	4,5	0	0	-	y	y	y	n	y	y	n	y	n	n	n	-	n	-	n	y	none
am04	music	64/128	1	*	0	0	-	y	y	y	n	n	y	n	y	n	n	n	-	y	-	n	n	none
am05	music	192/384	1	*	0	0	-	y	y	y	n	n	y	n	y	n	n	n	-	y	-	n	n	none
am06	music	128/256	1	*	0	0	-	y	y	y	n	n	y	n	y	n	n	n	-	y	-	n	n	none
am07	music	200/320	1	*	0	0	-	y	y	y	n	y	y	n	y	n	n	n	-	n	-	n	y	none
al00	sine sweep	40/64	2	*	0	0	-	-	-	y	n	n	n	n	-	n	n	n	-	n	-	n	n	RMS
al01	music	40/64	2	*	0	0	-	-	-	y	n	n	n	n	-	n	n	y	-	n	-	n	n	none
al02	music	40/64	2	*	0	0	-	-	-	y	n	n	n	n	-	n	n	y	-	n	-	y	n	none
al03	music	40/64	2	*	0	0	-	-	-	y	y	n	n	n	-	n	n	y	-	n	-	n	n	none
al04	music	40/64	2	*	0	0	-	-	-	y	n	y	y	y	-	n	n	y	-	n	-	n	n	none
al05	music	80/128	2	*	0	0	-	y	y	y	n	n	n	n	-	n	n	y	-	n	-	n	n	none
al06	test mix	120/192	2	0..2,7..11	0	0	-	y	y	y	n	n	y	n	-	n	n	y	-	n	-	n	n	none
al06	test mix	192	2	3..6	0	0	-	y	y	y	n	n	y	n	-	n	n	y	-	n	-	n	y	none
al07	music, other	240/384	2	*	0	0	-	y	y	y	n	n	y	n	-	n	n	n	-	n	-	n	y	none

file base name	content	bitrate (kbit/s)	AudioObjectType	SamplingFrequencyIndex	ChannelConfiguration	frameLengthFlag	coreCoderDelay	intensity	MS	window sequence switching	non-meaningful window _sequence transitions	window shape switching	tns_data_present	pulse data	prediction	LTP	PNS	data stream elements	gain compensation enabled	dynamic range control	bandwidth	buffer test	arithmetic torture	test procedure
al08	music	1920/ 3072	2	0..2,7..11	0	0	-	y	y	y	n	n	y	n	-	n	n	n	-	n	-	n	n	none
al08	music	3072	2	3..6	0	0	-	y	y	y	n	n	y	n	-	n	n	n	-	n	-	n	y	none
al14	music	64/128	2	*	0	0	-	y	y	y	n	n	y	n	-	n	n	n	-	y	-	n	n	none
al16	music	96/384	2	*	0	0	-	y	y	y	n	n	y	n	-	n	n	n	-	y	-	n	n	none
al17	music	80/128	2	*	0	0	-	n	n	y	n	n	n	n	-	n	n	n	-	n	-	n	n	none
al18	noise	40/64	2	*	1	0	-	-	-	l	n	n	n	n	-	n	y	n	-	n	-	n	n	PNS-1
al19	noise	40/64	2	*	1	0	-	-	-	s	n	n	n	n	-	n	y	n	-	n	-	n	n	PNS-2/3
as00	sine sweep	40/64	3	*	0	0	-	-	-	y	n	n	n	n	-	n	n	n	n	n	-	y	n	RMS
as01	music	40/64	3	*	0	0	-	-	-	y	n	y	y	n	-	n	n	n	y	n	4	n	n	none
as02	music	40/64	3	0..5, 8	0	0	-	-	-	y	n	y	y	n	-	n	n	n	y	n	3	n	n	none
as02	music	40/64	3	6, 7, 9..11	0	0	-	-	-	y	n	y	y	y	-	n	n	n	y	n	3	y	n	none
as03	music	40/64	3	0..9	0	0	-	-	-	y	n	y	y	n	-	n	n	n	y	n	2	n	n	none
as03	music	40	3	10, 11	0	0	-	-	-	y	n	y	y	y	-	n	n	n	y	n	2	y	n	none
as04	music	40/64	3	*	0	0	-	-	-	y	n	y	y	y	-	n	n	n	n	n	1	y	n	none
as05	music	40/64	3	0..4,6..11	0	0	-	-	-	y	n	n	n	n	-	n	n	y	n	n	4	n	n	none
as05	music	64	3	5	0	0	-	-	-	y	n	n	n	n	-	n	n	y	n	n	4	n	y	none
as06	music	40/64	3	0..4,6..11	0	0	-	-	-	y	n	n	n	n	-	n	n	y	n	n	3	n	n	none
as06	music	64	3	5	0	0	-	-	-	y	n	n	n	n	-	n	n	y	n	n	3	n	y	none
as07	music	40/64	3	0..3,5..11	0	0	-	-	-	y	n	n	n	n	-	n	n	y	n	n	2	n	n	none
as07	music	64	3	4	0	0	-	-	-	y	n	n	n	n	-	n	n	y	n	n	2	n	y	none
as08	music	40/64	3	*	0	0	-	-	-	y	n	n	n	n	-	n	n	y	n	n	1	n	n	none
as09	music	80/128	3	0..2,5..11	0	0	-	y	y	y	n	n	n	n	-	n	n	n	y	n	4	n	n	none
as09	music	128	3	3,4	0	0	-	y	y	y	n	n	n	n	-	n	n	n	y	n	4	n	y	none
as10	music	80/128	3	*	0	0	-	y	y	y	n	n	n	n	-	n	n	n	y	n	3	n	n	none
as11	music	80/128	3	*	0	0	-	y	y	y	n	n	n	n	-	n	n	n	y	n	2	n	n	none
as12	music	80/128	3	*	0	0	-	y	y	y	n	n	n	n	-	n	n	n	n	n	1	n	n	none
as13	music	320	3	0,1	0	0	-	y	y	y	n	y	y	y	-	n	n	n	y	n	4	n	n	none
as13	music	200/320	3	2..11	0	0	-	y	y	y	n	y	y	y	-	n	n	n	y	n	4	n	y	none
as14	music	200/320	3	0,9..11	0	0	-	y	y	y	n	y	y	y	-	n	n	n	y	n	3	n	n	none
as14	music	200/320	3	1..8	0	0	-	y	y	y	n	y	y	y	-	n	n	n	y	n	3	n	y	none
as15	music	280/448	3	0	0	0	-	y	y	y	n	y	y	y	-	n	n	n	y	n	4	n	n	none
as15	music	280/448	3	1..11	0	0	-	y	y	y	n	y	y	y	-	n	n	n	y	n	4	n	y	none
as16	music	280/448	3	0..8	0	0	-	y	y	y	n	y	y	y	-	n	n	n	y	n	3	n	y	none
as16	music	280	3	9..11	0	0	-	y	y	y	n	y	y	y	-	n	n	n	y	n	3	n	n	none
as17	music	40/64	3	*	0	0	-	-	-	y	y	n	n	n	-	n	n	n	y	n	4	n	n	none
ap01	sine sweep	64	4	3	0	0	-	-	-	y	n	y	n	n	-	y	n	n	-	n	-	n	n	RMS
ap02	music	128	4	3	0	0	-	n	y	y	n	y	y	n	-	y	n	n	-	n	-	n	n	none
ap03	music	64	4	3	0	0	-	-	-	y	n	y	y	n	-	y	n	n	-	n	-	n	n	none
ap04	music	64	4	3	0	0	-	-	-	y	n	y	n	n	-	y	n	n	-	n	-	n	n	none
ap05	music	128	4	3	0	0	-	y	y	y	n	y	y	n	-	y	n	n	-	n	-	n	n	none
ac01	test mix	32	6	3	1	0	-	-	-	y	n	y	?	n	-	y	n	-	-	n	-	n	n	none
		16	6	3	1	0	-	-	-	-	-	-	-	n	-		n	-	-	n	-	n	n	
		16	6	3	1	0	-	-	-	-	-	-	-	n	-		n	-	-	n	-	n	n	
		16	6	3	1	0	-	-	-	-	-	-	-	n	-		n	-	-	n	-	n	n	
		16	6	3	1	0	-	-	-	-	-	-	-	n	-		n	-	-	n	-	n	n	
		16	6	3	1	0	-	-	-	-	-	-	-	n	-		n	-	-	n	-	n	n	
		16	6	3	1	0	-	-	-	-	-	-	-	n	-		n	-	-	n	-	n	n	
		16	6	3	1	0	-	-	-	-	-	-	-	n	-		n	-	-	n	-	n	n	

file base name	content	bitrate (kbit/s)	AudioObjectType	SamplingFrequencyIndex	ChannelConfiguration	frameLengthFlag	coreCoderDelay	intensity	MS	window sequence switching	non-meaningful window _sequence transitions	window shape switching	tns_data_present	pulse data	prediction	LTP	PNS	data stream elements	gain compensation enabled	dynamic range control	bandwidth	buffer test	arithmetic torture	test procedure
ac02	test mix	32	6	3	1	0	-	-	-	y	n	y	?	n	-	y	n	-	-	n	-	n	n	none
		16	6	3	1	0	-	-	-	-	-	-	-	n	-	-	n	-	-	n	-	n	n	
		16	6	3	1	0	-	-	-	-	-	-	-	n	-	-	n	-	-	n	-	n	n	
		16	6	3	1	0	-	-	-	-	-	-	-	n	-	-	n	-	-	n	-	n	n	
		32	6	3	2	0	-	n	?	-	-	-	n	n	-	-	n	-	-	n	-	n	n	
		32	6	3	2	0	-	n	?	-	-	-	-	n	-	-	n	-	-	n	-	n	n	
		32	6	3	2	0	-	n	?	-	-	-	-	n	-	-	n	-	-	n	-	n	n	
		32	6	3	2	0	-	n	?	-	-	-	-	n	-	-	n	-	-	n	-	n	n	
ac03	test mix	64	6	3	2	0	-	-	-	y	n	y	?	n	-	y	n	-	-	n	-	n	n	none
		32	6	3	2	0	-	-	-	-	-	-	-	n	-	-	n	-	-	n	-	n	n	
		32	6	3	2	0	-	-	-	-	-	-	-	n	-	-	n	-	-	n	-	n	n	
		32	6	3	2	0	-	-	-	-	-	-	-	n	-	-	n	-	-	n	-	n	n	
		32	6	3	2	0	-	n	?	-	-	-	-	n	-	-	n	-	-	n	-	n	n	
		32	6	3	2	0	-	n	?	-	-	-	-	n	-	-	n	-	-	n	-	n	n	
		32	6	3	2	0	-	n	?	-	-	-	-	n	-	-	n	-	-	n	-	n	n	
		32	6	3	2	0	-	n	?	-	-	-	-	n	-	-	n	-	-	n	-	n	n	
ac04	test mix	6	8	12	1	-	-	-	-	-	-	-	-	n	-	-	n	-	-	-	-	-	-	none
		32	6	4	1	1	8000	-	-	y	n	y	?	n	-	-	n	-	-	n	-	n	n	
		32	6	4	1	1	-	-	-	-	-	-	-	n	-	-	n	-	-	n	-	n	n	
		32	6	4	1	1	-	-	-	-	-	-	-	n	-	-	n	-	-	n	-	n	n	
		32	6	4	2	1	-	n	?	-	-	-	n	n	-	-	n	-	-	n	-	n	n	
		32	6	4	2	1	-	n	?	-	-	-	-	n	-	-	n	-	-	n	-	n	n	
		32	6	4	2	1	-	n	?	-	-	-	-	n	-	-	n	-	-	n	-	n	n	
		32	6	4	2	1	-	n	?	-	-	-	-	n	-	-	n	-	-	n	-	n	n	
ac05	test mix	12.2	8	11	1	-	-	-	-	-	-	-	-	n	-	-	n	-	-	-	-	-	-	none
		2	8	11	1	1	-	-	-	-	-	-	-	n	-	-	n	-	-	-	-	-	-	
		16	6	3	1	1	0	-	-	y	n	y	?	n	-	-	n	-	-	n	-	n	n	
		16	6	3	1	1	-	-	-	-	-	-	-	n	-	-	n	-	-	n	-	n	n	
		16	6	3	1	1	-	n	?	-	-	-	n	n	-	-	n	-	-	n	-	n	n	
		32	6	3	2	1	-	n	?	-	-	-	-	n	-	-	n	-	-	n	-	n	n	
		32	6	3	2	1	-	n	?	-	-	-	-	n	-	-	n	-	-	n	-	n	n	
		32	6	3	2	1	-	n	?	-	-	-	-	n	-	-	n	-	-	n	-	n	n	
		32	6	3	2	1	-	n	?	-	-	-	-	n	-	-	n	-	-	n	-	n	n	
ac06	test mix	12.2	8	12	1	-	-	-	-	-	-	-	-	n	-	-	n	-	-	-	-	-	-	none
		2	8	12	1	1	-	-	-	-	-	-	-	n	-	-	n	-	-	-	-	-	-	
		32	6	7	2	1	798	n	?	y	n	y	?	n	-	-	n	-	-	n	-	n	n	
		32	6	7	2	1	-	n	?	-	-	-	-	n	-	-	n	-	-	n	-	n	n	
		32	6	7	2	1	-	n	?	-	-	-	-	n	-	-	n	-	-	n	-	n	n	
		32	6	7	2	1	-	n	?	-	-	-	-	n	-	-	n	-	-	n	-	n	n	
ac08	test mix	16	7	3	1	0	-	-	-	y	n	y	n	n	-	y	n	-	-	-	-	-	-	none
		16	6	3	1	0	-	-	-	-	-	-	y	n	-	-	n	-	-	n	-	n	n	
		16	6	3	1	0	-	-	-	-	-	-	-	n	-	-	n	-	-	n	-	n	n	
		16	6	3	1	0	-	-	-	-	-	-	-	n	-	-	n	-	-	n	-	n	n	
		16	6	3	1	0	-	-	-	-	-	-	-	n	-	-	n	-	-	n	-	n	n	
		16	6	3	1	0	-	-	-	-	-	-	-	n	-	-	n	-	-	n	-	n	n	
		16	6	3	1	0	-	-	-	-	-	-	-	n	-	-	n	-	-	n	-	n	n	
		16	6	3	1	0	-	-	-	-	-	-	-	n	-	-	n	-	-	n	-	n	n	
		16	6	3	1	0	-	-	-	-	-	-	-	n	-	-	n	-	-	n	-	n	n	

file base name	content	bitrate (kbit/s)	AudioObjectType	SamplingFrequencyIndex	ChannelConfiguration	frameLengthFlag	coreCoderDelay	intensity	MS	window sequence switching	non-meaningful window _sequence transitions	window shape switching	tns_data_present	pulse data	prediction	LTP	PNS	data stream elements	gain compensation enabled	dynamic range control	bandwidth	buffer test	arithmetic torture	test procedure
ac09	test mix	16	7	3	1	0	-	-	-	y	n	y	n	n	-	n	n	-	-	-	-	-	-	none
		16	6	3	1	0	-	-	-	-	-	-	y	n	-	-	n	-	-	n	-	n	n	
		16	6	3	1	0	-	-	-	-	-	-	-	n	-	-	n	-	-	n	-	n	n	
		16	6	3	1	0	-	-	-	-	-	-	-	n	-	-	n	-	-	n	-	n	n	
		16	6	3	1	0	-	-	-	-	-	-	-	n	-	-	n	-	-	n	-	n	n	
		32	6	3	2	0	-	n	?	-	-	-	y	n	-	-	n	-	-	n	-	n	n	
		32	6	3	2	0	-	n	?	-	-	-	-	n	-	-	n	-	-	n	-	n	n	
		32	6	3	2	0	-	n	?	-	-	-	-	n	-	-	n	-	-	n	-	n	n	
		32	6	3	2	0	-	n	?	-	-	-	-	n	-	-	n	-	-	n	-	n	n	
ac10	test mix	16	7	7	1	0	-	-	-	y	n	y	n	n	-	n	n	-	-	-	-	-	-	none
		32	6	7	2	0	-	n	?	-	-	-	y	n	-	-	n	-	-	n	-	n	n	
		32	6	7	2	0	-	n	?	-	-	-	-	n	-	-	n	-	-	n	-	n	n	
		32	6	7	2	0	-	n	?	-	-	-	-	n	-	-	n	-	-	n	-	n	n	
		32	6	7	2	0	-	n	?	-	-	-	-	n	-	-	n	-	-	n	-	n	n	
ac12	test mix	6	8	11	1	1	-	-	-	-	-	-	-	n	-	-	n	-	-	-	-	-	-	none
		16	6	11	1	1	946	n	?	y	n	y	y	?			n	-	-	n	-	n	n	
ac13	test mix	6	8	11	1	1	-	-	-	-	-	-	-	n	-	-	n	-	-	-	-	-	-	none
		32	6	6	1	1	798	-	-	y	n	y	y	n			n	-	-	n	-	n	n	
ac14	test mix	6	8	11	1	1	-	-	-	-	-	-	-	n	-	-	n	-	-	-	-	-	-	none
		64	6	5	2	1	404	n	?	y	n	y	?	n			n	-	-	n	-	n	n	
		128	6	5	2	1	-	n	?	-	-	-	-	n			n	-	-	n	-	n	n	
ac15	test mix	6	8	11	1	1	-	-	-	-	-	-	-	n	-	-	n	-	-	-	-	-	-	none
		64	6	0	1	1	0	-	?	y	n	y	?	n			n	-	-	n	-	n	n	
		128	6	0	2	1	-	n	?	-	-	-	n	n			n	-	-	n	-	n	n	
ac16	test mix	8	7	9	1	0	-	-	-	y	n	y	n	n	-	y	n	-	-	-	-	-	-	none
		16	6	9	1	0	-	-	-	-	-	-	?	n			n	-	-	n	-	n	n	
		32	6	9	2	0	-	n	?	-	-	-	n	n			n	-	-	n	-	n	n	
ac17	test mix	16	7	6	1	0	-	-	-	y	n	y	n	n	-	y	n	-	-	-	-	-	-	none
		32	6	6	1	0	-	-	-	-	-	-	?	n			n	-	-	n	-	n	n	
		64	6	6	2	0	-	n	?	-	-	-	n	n			n	-	-	n	-	n	n	
ac18	test mix	16	7	5	1	0	-	-	-	y	n	y	n	n	-	y	n	-	-	-	-	-	-	none
		64	6	5	1	0	-	-	-	-	-	-	?	n			n	-	-	n	-	n	n	
		128	6	5	2	0	-	n	?	-	-	-	n	n			n	-	-	n	-	n	n	
ac19	test mix	32	7	2	1	0	-	-	-	y	n	y	n	n	-	y	n	-	-	-	-	-	-	none
		96	6	2	1	0	-	-	-	-	-	-	?	n			n	-	-	n	-	n	n	
		192	6	2	2	0	-	n	?	-	-	-	n	n			n	-	-	n	-	n	n	
ac20	test mix	16	7	1	1	1	-	-	-	y	n	y	n	n	-	n	n	-	-	-	-	-	-	none
		64	6	1	2	1	-	n	y	-	-	-	y	n			n	-	-	n	-	n	n	
		64	6	1	2	1	-	n	y	-	-	-	-	n			n	-	-	n	-	n	n	
		64	6	1	2	1	-	n	y	-	-	-	-	n			n	-	-	n	-	n	n	
		64	6	1	2	1	-	n	y	-	-	-	-	n			n	-	-	n	-	n	n	
ac22	test mix	32	6	3	2	0	-	y	y	y	n	y	y	n	-	y	n	-	-	n	-	n	n	none
		32	6	3	2	0	-	y	y	-	-	-	-	n			n	-	-	n	-	n	n	
		32	6	3	2	0	-	n	y	-	-	-	-	n			n	-	-	n	-	n	n	
		32	6	3	2	0	-	n	y	-	-	-	-	n			n	-	-	n	-	n	n	

Table 14 b) — AAC test sequences with a framelength 960 samples

file base name	content	lower bitrate (kbit/s)	higher bitrate (kbit/s)	AudioObjectType	SamplingFrequencyIndex	ChannelConfiguration	corresponding sequence in terms of properties	frameLengthFlag	test procedure
al00sf	sine sweep	43	69	2	*	0	al00	1	RMS
al01sf	music	43	69	2	*	0	al01	1	none
al02sf	music	43	69	2	*	0	al02	1	none
al03sf	music	43	69	2	*	0	al03	1	none
al04sf	music	43	69	2	*	0	al04	1	none
al05sf	music	86	137	2	*	0	al05	1	none
al06sf	test mix	128	205	2	0..2,7..11	0	al06	1	none
al06sf	test mix	205	205	2	3..6	0	al06	1	none
al07sf	music, other	256	410	2	*	0	al07	1	none
al08sf	music	2048	3277	2	0..2,7..11	0	al08	1	none
al08sf	music	3277	3277	2	3..6	0	al08	1	none
al14sf	music	69	137	2	*	0	al14	1	none
al15sf	music	205	410	2	*	0	al15	1	none
al16sf	music	103	410	2	*	0	al16	1	none
al17sf	music	86	137	2	*	0	al17	1	none
al18sf	noise	43	69	2	*	1	al18	1	PNS-1
al19sf	noise	43	69	2	*	1	al19	1	PNS-2/3

Table 15 — ER AAC test sequences

file base name	content	bitrate (kbit/s)	AudioObjectType	SamplingFrequencyIndex	ChannelConfiguration	epConfig	frameLengthFlag	coreCoderDelay	aacSectionDataResilienceFlag	aacScalefactorDataResilienceFlag	aacSpectralDataResilienceFlag	intensity	MS	window sequence switching	window shape switching	TNS	pulse data	LTP	PNS	test procedure
er_al10	sine sweep	40/64	17	*	1	0,1	0	-	0	0	0	-	-	y	y	n	n	n	n	RMS
er_al12	test mix	40/64	17	*	1	0,1	0	-	1	0	0	-	-	?	?	?	n	n	fs1	none
er_al15	test mix	40/64	17	*	1	0,1	0	-	0	0	1	-	-	?	?	?	n	n	fs2	none
er_al18	test mix	40/64	17	*	1	0,1	0	-	1	1	1	-	-	?	?	?	n	n	fs1	none
er_al21	test mix	80/128	17	*	2	0,1	0	-	0	0	0	y	y	?	?	?	n	n	fs2	none
er_al23	test mix	80/128	17	*	2	0,1	0	-	0	1	0	y	y	?	?	?	n	n	fs1	none
er_al26	test mix	80/128	17	*	2	0,1	0	-	1	0	1	y	y	?	?	?	n	n	fs2	none
er_ap10	sine sweep	40/64	19	*	1	0,1	0	-	0	0	0	-	-	y	y	n	n	?	n	RMS
er_ap14	test mix	40/64	19	*	1	0,1	0	-	1	1	0	-	-	?	?	?	n	?	fs1	none
er_ap27	test mix	80/128	19	*	2	0,1	0	-	0	1	1	y	y	?	?	?	n	?	fs2	none
er_ad1000	sine sweep	64	23	*	1	0,1	0	-	0	0	0	-	-	y	n	n	?	n	RMS	
er_ad1020	test mix	64	23	*	1	0,1	0	-	0	1	0	-	-	-	?	?	n	?	fs1	none
er_ad1030	test mix	64	23	*	1	0,1	0	-	0	1	1	-	-	-	?	?	n	?	fs2	none
er_ad1109	noise	64	23	*	1	0,1	1	-	0	0	0	-	-	-	n	n	n	n	y	PNS-1
er_ad1102	sine sweep	64	23	*	1	0,1	1	-	0	0	0	-	-	-	y	n	n	?	n	RMS
er_ad1103	test mix	64	23	*	1	0,1	1	-	0	0	0	-	-	-	?	?	n	?	fs2	none
er_ad1111	test mix	64	23	*	1	0,1	1	-	0	0	1	-	-	-	?	?	n	?	fs1	none
er_ad2040	test mix	128	23	*	2	0,1	0	-	1	0	0	y	y	-	?	?	n	?	fs1	none
er_ad2050	test mix	128	23	*	2	0,1	0	-	1	0	1	y	y	-	?	?	n	?	fs2	none
er_ad2160	test mix	128	23	*	2	0,1	1	-	1	1	0	y	y	-	?	?	n	?	fs1	none
er_ad2170	test mix	128	23	*	2	0,1	1	-	1	1	1	y	y	-	?	?	n	?	fs2	none
er_ad1000np	sine sweep	64	23	*	1	0,1	0	-	0	0	0	-	-	-	y	n	n	n	n	RMS
er_ad1020np	test mix	64	23	*	1	0,1	0	-	0	1	0	-	-	-	?	?	n	n	fs2	none
er_ad1030np	test mix	64	23	*	1	0,1	0	-	0	1	1	-	-	-	?	?	n	n	fs1	none
er_ad1102np	sine sweep	64	23	*	1	0,1	1	-	0	0	0	-	-	-	y	n	n	n	n	RMS
er_ad1103np	test mix	64	23	*	1	0,1	1	-	0	0	0	-	-	-	?	?	n	n	fs1	none
er_ad1111np	test mix	64	23	*	1	0,1	1	-	0	0	1	-	-	-	?	?	n	n	fs2	none
er_ad2040np	test mix	128	23	*	2	0,1	0	-	1	0	0	y	y	-	?	?	n	n	fs2	none
er_ad2050np	test mix	128	23	*	2	0,1	0	-	1	0	1	y	y	-	?	?	n	n	fs1	none
er_ad2160np	test mix	128	23	*	2	0,1	1	-	1	1	0	y	y	-	?	?	n	n	fs2	none
er_ad2170np	test mix	128	23	*	2	0,1	1	-	1	1	1	y	y	-	?	?	n	n	fs1	none
er_ad1100np_ep0	sine sweep	64	23	3,4,5,6,7	1	0	1	-	0	0	0	n	n	-	n	n	-	-	n	RMS
er_ad1009np_ep0	noise	64	23	3,4,5,6,7	1	0	0	-	0	0	0	n	n	-	n	n	-	-	y	PNS-1
er_ad1110np_ep0	test mix	64	23	3,4,5,6,7	1	0	1	-	0	0	1	n	n	-	n	n	-	-	n	RMS
er_ad1120np_ep0	test mix	64	23	3,4,5,6,7	1	0	1	-	0	1	0	n	n	-	n	n	-	-	n	RMS
er_ad1140np_ep0	test mix	64	23	3,4,5,6,7	1	0	1	-	1	0	0	n	n	-	n	n	-	-	n	RMS
er_ad1130np_ep0	test mix	64	23	3,4,5,6,7	1	0	1	-	0	1	1	n	n	-	n	n	-	-	n	RMS
er_ad1150np_ep0	test mix	64	23	3,4,5,6,7	1	0	1	-	1	0	1	n	n	-	n	n	-	-	n	RMS
er_ad1160np_ep0	test mix	64	23	3,4,5,6,7	1	0	1	-	1	1	0	n	n	-	n	n	-	-	n	RMS
er_ad1170np_ep0	test mix	64	23	3,4,5,6,7	1	0	1	-	1	1	1	n	n	-	n	n	-	-	n	RMS
er_eld1000np_ep0	sine sweep	64	39	3,4,5,6,7	1	0	0	-	0	0	0	n	n	-	n	n	-	-	n	RMS k=TBD

file base name	content	bitrate (kbit/s)	AudioObjectType	SamplingFrequencyIndex	ChannelConfiguration	epConfig	frameLengthFlag	coreCoderDelay	aacSectionDataResilienceFlag	aacScalefactorDataResilienceFla g	aacSpectralDataResilienceFlag	intensity	MS	window sequence switching	window shape switching	TNS	pulse data	LTP	PNS	test procedure
er_eld1100np_ep0	sine sweep	64	39	3,4,5,6,7	1	0	1	-	0	0	0	n	n	-	n	n	-	-	n	RMS k=TBD
er_eld1101np_ep0	test mix	64	39	3,4,5,6,7	1	0	1	-	0	0	0	n	n	-	n	y	-	-	n	none k=TBD
er_eld1001np_ep0	test mix	64	39	3,4,5,6,7	1	0	0	-	0	0	0	n	n	-	n	y	-	-	n	none k=TBD
er_eld1102np_ep0	noise	64	39	3,4,5,6,7	1	0	1	-	0	0	0	n	n	-	n	n	-	-	y	PNS-1
er_eld1002np_ep0	noise	64	39	3,4,5,6,7	1	0	0	-	0	0	0	n	n	-	n	n	-	-	y	PNS-1
er_eld2100np_ep0	test mix	64	39	3,4,5,6,7	2	0	1	-	0	0	0	n	y	-	n	n	-	-	n	RMS k=TBD
er_eld2000np_ep0	test mix	64	39	3,4,5,6,7	2	0	0	-	0	0	0	n	y	-	n	n	-	-	n	RMS k=TBD
er_eld1110np_ep0	test mix	64	39	3,4,5,6,7	1	0	1	-	0	0	1	n	n	-	n	n	-	-	n	RMS k=TBD
er_eld1120np_ep0	test mix	64	39	3,4,5,6,7	1	0	1	-	0	1	0	n	n	-	n	n	-	-	n	RMS k=TBD
er_eld1140np_ep0	test mix	64	39	3,4,5,6,7	1	0	1	-	1	0	0	n	n	-	n	n	-	-	n	RMS k=TBD
er_eld1130np_ep0	test mix	64	39	3,4,5,6,7	1	0	1	-	0	1	1	n	n	-	n	n	-	-	n	RMS k=TBD
er_eld2150np_ep0	test mix	64	39	3,4,5,6,7	2	0	1	-	1	0	1	n	y	-	n	n	-	-	n	RMS k=TBD
er_eld1160np_ep0	test mix	64	39	3,4,5,6,7	1	0	1	-	1	1	0	n	n	-	n	n	-	-	n	RMS k=TBD
er_eld1170np_ep0	test mix	64	39	3,4,5,6,7	1	0	1	-	1	1	1	n	n	-	n	n	-	-	n	RMS k=TBD
er_eld1010np_ep0	test mix	64	39	3,4,5,6,7	1	0	0	-	0	0	1	n	n	-	n	n	-	-	n	RMS k=TBD
er_eld1020np_ep0	test mix	64	39	3,4,5,6,7	1	0	0	-	0	1	0	n	n	-	n	n	-	-	n	RMS k=TBD
er_eld1040np_ep0	test mix	64	39	3,4,5,6,7	1	0	0	-	1	0	0	n	n	-	n	n	-	-	n	RMS k=TBD
er_eld1030np_ep0	test mix	64	39	3,4,5,6,7	1	0	0	-	0	1	1	n	n	-	n	n	-	-	n	RMS k=TBD
er_eld1050np_ep0	test mix	64	39	3,4,5,6,7	1	0	0	-	1	0	1	n	n	-	n	n	-	-	n	RMS k=TBD
er_eld1060np_ep0	Test mix	64	39	3,4,5,6,7	1	0	0	-	1	1	0	n	n	-	n	n	-	-	n	RMS k=TBD
er_eld2070np_ep0	Test mix	64	39	3,4,5,6,7	2	0	0	-	1	1	1	n	y	-	n	n	-	-	n	RMS k=TBD
er_ac111	test mix	64 64	20 20	3 3	1 2	0,1	0 0	- -	0 0	0 0	0 1	- n	- y	? -	? -	? ?	n n	n -	n n	none

file base name	content	bitrate (kbit/s)	AudioObjectType	SamplingFrequencyIndex	ChannelConfiguration	epConfig	frameLengthFlag	coreCoderDelay	aacSectionDataResilienceFlag	aacScalefactorDataResilienceFlag	aacSpectralDataResilienceFlag	intensity	MS	window sequence switching	window shape switching	TNS	pulse data	LTP	PNS	test procedure
er_ac118	sine sweep	40	20	2	1	0,1	0	-	0	1	0	-	-	y	y	n	n	n	n	RMS
		24	20	2	1		0	-	0	1	1	-	-	-	-	-	n	-	n	
		40	20	2	2		0	-	1	0	0	n	y	-	-	n	n	-	n	
		40	20	2	2		0	-	1	0	1	n	y	-	-	-	n	-	n	
er_ac119	sine sweep	30	20	7	1	0,1	0	-	0	1	0	-	-	y	y	n	n	n	n	RMS
		16	20	7	1		0	-	0	1	1	-	-	-	-	-	n	-	n	
		96	20	7	2		0	-	1	0	0	n	y	-	-	n	n	-	n	
		20	20	7	2		0	-	1	0	1	n	y	-	-	-	n	-	n	
er_ac121	music	32	20	3	2	0	0	-	0	0	0	y	y	n	n	n	n	-	n	none
		32	20	3	2		0	-	0	0	0	y	y	-	-	-	n	-	n	
		32	20	3	2		0	-	0	0	0	n	y	-	-	-	n	-	n	
		32	20	3	2		0	-	0	0	0	n	y	-	-	-	n	-	n	
er_ac123	test mix	40	20	4	2	0,1	0	-	1	1	0	n	y	?	?	?	n	n	y	none
		64	20	4	2		0	-	1	1	1	n	y	-	-	-	n	-	n	
er_ac211	test mix	6	24	11	1	0,1	-	-	-	-	-	-	-	-	-	-	-	-	-	none
		40	20	6	1		1	0	0	-	0	-	-	?	?	?	n	-	y	
		64	20	6	2		1	-	0	0	1	n	y	-	-	?	n	-	n	
er_ac221	test mix	6.2	24	12	1	0,1	-	-	-	-	-	-	-	-	-	-	-	-	-	none
		64	20	7	2		1	0	0	0	0	n	y	?	?	?	n	-	n	
		128	20	7	2		1	-	0	1	1	n	y	-	-	-	n	-	n	
er_ac311	test mix	8	21	5	1	0,1	0	-	-	-	-	-	-	?	?	-	-	n	-	none
		40	20	5	1		0	-	1	0	0	-	-	-	-	?	n	-	n	
		64	20	5	2		0	-	1	1	1	n	y	-	-	?	n	-	n	
er_ac321	test mix	12	21	6	1	0,1	1	-	-	-	-	-	-	?	?	-	-	n	-	none
		64	20	6	2		1	-	1	1	0	n	y	-	-	?	n	-	y	
		96	20	6	2		1	-	1	0	1	n	y	-	-	-	n	-	n	

Table 16 — Specificaiton of the program_config_element()

file name	num_front_channel_elements	front_element_is_cpe	num_side_channel_elements	side_element_is_cpe	num_back_channel_elements	back_element_is_cpe	num_lfe_channel_elements	num_valid_cc_elements	cc_element_is_ind_sw
am00	1	n	0	-	0	-	0	0	-
am01	1	n	0	-	0	-	0	0	-
am02	1	y	0	-	0	-	0	0	-
am04	1	y	0	-	0	-	0	0	-
am05	2	n y	0	-	1	y	1	1	y
am06	2	n y	0	-	0	-	0	0	-
am07	2	n y	0	-	1	y	1	0	-
al00	1	n	0	-	0	-	0	0	-
al01	1	n	0	-	0	-	0	0	-
al02	1	n	0	-	0	-	0	0	-
al03	1	n	0	-	0	-	0	0	-
al04	1	n	0	-	0	-	0	0	-
al05	1	y	0	-	0	-	0	0	-
al06	2	n y	0	-	0	-	0	0	-
al07	2	n y	0	-	1	y	1	1	n
al08	11	n n n n n n n y y y y	9	y y y y y y y y y	12	y y y n n n n n n n n n	0	0	-
al14	1	y	0	-	0	-	0	0	-
al16	2	n y	0	-	0	-	0	0	-
al17	2	n n	0	-	0	-	0	0	-
as00	1	n	0	-	0	-	0	0	-
as01	1	n	0	-	0	-	0	0	-
as02	1	n	0	-	0	-	0	0	-
as03	1	n	0	-	0	-	0	0	-
as04	1	n	0	-	0	-	0	0	-
as05	1	n	0	-	0	-	0	0	-
as06	1	n	0	-	0	-	0	0	-
as07	1	n	0	-	0	-	0	0	-
as08	1	n	0	-	0	-	0	0	-
as09	1	y	0	-	0	-	0	0	-
as10	1	y	0	-	0	-	0	0	-
as11	1	y	0	-	0	-	0	0	-
as12	1	y	0	-	0	-	0	0	-
as13	2	n y	1	n	0	-	1	0	-

file name	num_front_channel_elements	front_element_is_cpe	num_side_channel_elements	side_element_is_cpe	num_back_channel_elements	back_element_is_cpe	num_lfe_channel_elements	num_valid_cc_elements	cc_element_is_ind_sw
as14	2	n y	1	n	0	-	1	0	-
as15	3	n y y	0	-	1	n	1	0	-
as16	3	n y y	0	-	1	n	1	0	-
as17	1	n	0	-	0	-	0	0	-
ap01	1	n	0	-	0	-	0	0	-
ap02	2	n	0	-	0	-	0	0	-
ap03	1	n	0	-	0	-	0	0	-
ap04	1	n	0	-	0	-	0	0	-
ap05	2	n	0	-	0	-	0	0	-

Legend:
"*" – variable
"?" – might be used
"-" – not applicable
"fs_1" – yes if fs is one of the following: 08, 12, 22, 32, 48, 88; no otherwise
"fs_2" – yes if fs is one of the following: 11, 16, 24, 44, 64, 96; no otherwise
"l" – long
"n" – no
"s" – short
"y" – yes

7.5 TwinVQ and ER_TwinVQ

7.5.1 DecoderSpecificInfo Characteristics

Encoders may apply restrictions to the following parameters of the Object Descriptor Stream:

a) samplingFrequencyIndex (descriptor element which indicates sampling rate).
b) bitrate (indicates bitrate).
c) number of layers (indicates the number of scalable layers).
d) number of channels (indicates the number of channels of input signal).
e) frameLength (indicate frame length is 1024 or 960).

7.5.2 Audio Access Unit Characteristics

Encoders may apply restrictions to the following parameters of the bitstream:

a) window_sequence
b) window_shape
c) LTP (ltp_present)
d) M/S stereo (msmask_present)
e) TNS (tns_present)
f) quantizer option (bandlimit, ppc, postprocess)

7.5.3 Procedure to Test Bitstream Conformance

7.5.3.1 Parsing system layer parameters

The decoder shall get the information of the sampling frequency, number of layers, bitrate and number of channels from the system layer.

7.5.3.2 Decoding of the payload

7.5.3.2.1 parsing tvq_scalable_main_header()

The syntax has the **window_sequence**, **window_shape**, **ms_mask_present**, **scale_factor_grouping**, **ltp_data_present**, and **tns_data_resent**. These syntax elements are common to those for AAC.

7.5.3.2.2 parsing tvq_scalable_extension_header()

The syntax has **ms_mask_present** common to AAC.

7.5.3.2.3 parsing vq_single_element()

The syntax has the flags for the quantizer option of **band_limit**, **ppc**, **postprocess**, as well as the main quantization information. Detailed specification is described in ISO/IEC 14496-3 subpart 4.

7.5.4 Decoder Characteristics

A conformant decoder shall support all characteristics given by the definition of level in the scaleable profile.

7.5.5 Procedure to Test Decoder Conformance

For the purpose of testing the processing at the decoder, number of bitstreams and the associated reference output PCM signals are supplied as listed in Table 17, Table 18 and Table 19. They cover the wide range of sampling rate, bit rate, number of channels, number of scalable layers, AAC related tools (window_shape, LTP, M/S stereo, TNS), and TwinVQ specific quantizer options (bandlimit, ppc, postprocess).

TV20 and TV24 contain the code to scan all codebook tables of the vector quantizors.

The ms_mode 1 means that the **ms_mask_present** == 1 or 0, ms_mode_present 2 means that **ms_mask_present** == 2 or 0.

The actual bit rate of the bitstreams may be slightly less than the values listed due to the byte alignment process.

Two-step accuracy criteria for conformance

A two-step approach is used to distinguish between two levels of accuracy, namely Fixed-Point accuracy and full accuracy of accuracy for decoder conformance.

Full Accuracy: A decoder meeting the stronger Full Accuracy conformance requirements may be called a Full Accuracy conformant decoder. This level of accuracy is intended for decoders running on floating-point platforms, enabling higher-precision mathematical operations.

Fixed-Point Accuracy: A decoder may be called conformant with Fixed-Point Accuracy in case the Fixed-Point Accuracy conformance criteria are met. Decoders with a limited accuracy due to fixed-point internal calculations may use these conformance criteria to verify the validity of the decoder.

Conformance criterion for Full Accuracy TwinVQ and ER_TwinVQ decoders

The RMS/LSB Measurement test procedure applies (see 7.1.2.2.1)

Conformance criteria for Fixed-Point Accuracy TwinVQ and ER_TwinVQ decoders

The conformance criteria for Fixed-Point Accuracy decoders are based on measuring the segmental SNR and the LPC cepstral distortion (CD) between the Reference decoder output and the output of the decoder to be tested. The segment length to be used in the calculation of the SNR is equal to the general audio frame length, namely 1024 or 960. The SNR and the CD have to be calculated only for the segments of which the power of the Reference signal is in the range [-50...-15] dB. CD is defined as

$$CD = \frac{10}{\ln(10)} \cdot \sqrt{2D}$$

D is the accumulated distortion of the LPC cepstrum C_{ref} of the reference signal and C_{test} of the output of the decoder under test. D is defined as

$$D = \sum_{i=1}^{N} \left(C_{ref}[i] - C_{test}[i] \right)^2$$

N is the LPC cepstrum order which equals 32. The LPC cepstrum C[i] is defined by means of the algorithm `lpc2cepstrum` based on the LPC coefficients of a 16th order linear prediction filter. The computation of the LPC filter coefficients lpc_coef [j] is defined by the algorithm `calculate_lpc`.

To be called an ISO/IEC 14496-3 TwinVQ object type decoder with Fixed-Point Accuracy, the average value of the segmental SNR shall exceed 30 dB and at the same time the average value of the CD shall not exceed 1 dB.

7.5.6 Descriptions of the audio test bitstreams

Table 17 — TwinVQ Object Type Test Bitstreams

File Name	TV00	TV01	TV02	TV03	TV04	TV05	TV06	TV07
content	music	music	music	music	music	music	music	music
level in scalable profile	1	1	1	1	1	1	1	1
bitrate [kbit/s]	8	16	16	16	16	16	16	16
sampling rate [kHz]	8	16	16	16	16	16	16	16
frame length	1024	1024	1024	1024	960	1024	1024	1024
number of scaleable layers	1	1	1	1	1	1	1	1
number of channels	1	1	1	1	1	1	1	1
long-term prediction (LTP)	no	no	yes	no	no	no	no	no
adaptive window shape	no	yes	no	no	no	no	no	no
TNS	no	no	no	yes	no	no	no	no
M/S stereo mode 2	-	-	-	-	-	-	-	-
M/S stereo mode 1	-	-	-	-	-	-	-	-
bandlimit_present	no	no	no	no	no	yes	no	no
ppc_present	no	no	no	no	no	no	yes	no
postprocess_present	no	no	no	no	no	no	no	yes

Table 18 — TwinVQ Object Type Test Bitstreams (continued)

File Name	TV11	TV14	TV15	TV16	TV20	TV21
content	music	music	music	music	music	music
level in scalable profile	2	1	3	2	1	2
bitrate [kbit/s]	16 + 16	8+8+8	32+32+32	12*8	8	16
sampling rate [kHz]	16	24	48	44.1	16	44.1
frame length	1024	1024	1024	1024	1024	1024
number of scaleable layers	1	3	3	8	1	1
number of channels	2	1	2	1	1	1
long-term prediction (LTP)	no	no	no	no	no	no
adaptive window shape	no	no	no	no	no	yes
TNS	no	no	no	no	no	no
M/S stereo mode 2	yes	-	yes	-	-	-
M/S stereo mode 1	yes	-	no	-	-	-
bandlimit_present	no	no	no	no	no	yes
ppc_present	no	no	no	no	no	yes
postprocess_present	no	no	no	no	no	yes
					yes	no

Table 19 — TwinVQ Object Type Test Bitstreams (continued)

File Name	TV22	TV23	TV24	TV25	TV26	TV27
content	music	music	music	music	music	music
level in scalable profile	3	1	1	2	3	1
bitrate [kbit/s]	32	16	16+16	16+16+16	32+32	16+16+16
sampling rate [kHz]	48	24	16	32	32	22.05
frame length	1024	1024	1024	1024	1024	1024
number of scaleable layers	1	1	2	3	2	3
number of channels	2	1	1	1	2	1
long-term prediction (LTP)	no	yes	no	no	no	yes
adaptive window shape	no	no	no	yes	no	no
TNS	no	yes	no	no	no	yes
M/S stereo mode 2	yes	-	-	-	yes	-
M/S stereo mode 1	yes	-	-	-	yes	-
bandlimit_present	no	no	no	yes	no	no
ppc_present	no	no	no	yes	no	no
postprocess_present	no	no	no	yes	no	no
scan all codebook	no	no	yes	no	no	no

Table 20 — ER_TwinVQ Object Type Test Bitstreams

File base name	er_tv01	er_tv02
level in scalable profile	2	3
Bitrate per channel [kbit/s]	16	16+16+16
Sampling rate [kHz]	32	48
Frame length	1024	1024
Number of scaleable layers	1	3
Number of channels	2	1
long-term prediction (LTP)	-	-
Adaptive window shape	yes	no
TNS	no	yes
M/S stereo mode 2	yes	-
M/S stereo mode 1	yes	-
Bandlimit_present	yes	no
ppc_present	yes	no
Postprocess_present	yes	no

7.6 ER BSAC

The ER Fine Granue Audio Object Type is an extension of the AAC LC Object Type with a new noiseless coding scheme to support the fine grain scalability and error resilience. A Bit-sliced arithemtic coding replaces the Huffman Coding of AAC. This object type uses different compressed data syntax. The multichannel ER BSAC object is supported with BSAC channel extension payload.

7.6.1 Compressed data

7.6.1.1 Characteristics

7.6.1.1.1 AudioSpecificConfig

There are several constraints for the values of AudioSpecificConfig. An encoder may apply restrictions to the following parameters of the AudioSpecificConfig:

AudioObjectType

SamplingFrequencyIndex

SamplingFrequency (if SamplingFrequencyIndex = 0xf)

ChannelConfiguration

extensionFlag

7.6.1.1.2 Bitstream payload

These characteristics specify the constraints that are applied by the encoder in generating the Audio Access Units. Encoders may apply restrictions to the following parameters of the Audio Access Units:

use of long term prediction (LTP)

window_shape

M/S stereo

intensity stereo

TNS

segmented arithmetic coding(sba) mode

7.6.1.2 Test procedure

7.6.1.2.1 AudioSpecificConfig

The following restrictions apply to AudioSpecificConfig:

AudioObjectType: Shall be encoded with the value 22

SamplingFrequencyIndex: Shall be encoded with the following values:

Table 21

SamplingFrequencyIndex	Level 1	Level 2	Level 3	Level 4	Level 5	Level 6
Mobile Audio Internetworking Profile	>= 0x6	>= 0x03	>= 0x3	>= 0x6	>= 0x03	>= 0x03
Natural Audio Profile	>= 0x03				not used	

SamplingFrequency (if SamplingFrequencyIndex = 0xf): Shall be encoded with the following values:

Table 22

SamplingFrequency	Level 1	Level 2	Level 3	Level 4	Level 5	Level 6
Mobile Audio Internetworking Profile	<= 24000	<= 48000	<= 48000	<= 24000	<= 48000	<= 48000
Natural Audio Profile	<= 48000				not used	

ChannelConfiguration: Shall be encoded with the following values:

Table 23

ChannelConfiguration	Level 1	Level 2	Level 3	Level 4	Level 5	Level 6
Mobile Audio Internetworking Profile	1	1..2	1..2	1	1..2	1..2
Natural Audio Profile	1..2				not used	

The following restrictions apply to GASpecificConfig:

FrameLengthFlag: shall be encoded with the value 0

DependsOnCoreCoder: shall be encoded with the value 0

CoreCoderDelay: not applicable

ExtensionFlag: shall be encoded with the value 1

numOfSubFrame: shall be encoded with the value larger than 0

layer_length: shall be encoded with the value larger than 3

extensionFlag3: shall be encoded with the value 0

7.6.1.2.2 Audio Acess Units

7.6.1.2.2.1 bsac_base_element ()

frame_length: must be larger than or equal to 4

7.6.1.2.2.2 bsac_header ()

header_length: ((header_length+7)*8) must be smaller than or equal to (frame_length*8)

top_layer: must be larger than or equal to (bitrate/1000/nch).

base_band: must be larger than 0.

7.6.1.2.2.3 general_header ()

reserved_bit: must be set to zero.

window_sequence: The meaningful window_sequence transitions are as follows:

from ONLY_LONG_SEQUENCE to {ONLY_LONG_SEQUENCE, LONG_START_SEQUENCE}

from LONG_START_SEQUENCE to {EIGHT_SHORT_SEQUENCE, LONG_STOP_SEQUENCE}

from LONG_STOP_SEQUENCE to {ONLY_LONG_SEQUENCE, LONG_START_SEQUENCE}

from EIGHT_SHORT_SEQUENCE to {EIGHT_SHORT_SEQUENCE, LONG_STOP_SEQUENCE}

Other, non-meaningful, window_sequence transitions are also possible:

from ONLY_LONG_SEQUENCE to {EIGHT_SHORT_SEQUENCE, LONG_STOP_SEQUENCE}

from LONG_START_SEQUENCE to {ONLY_LONG_SEQUENCE, LONG_START_SEQUENCE}

from LONG_STOP_SEQUENCE to {EIGHT_SHORT_SEQUENCE, LONG_STOP_SEQUENCE}

from EIGHT_SHORT_SEQUENCE to {ONLY_LONG_SEQUENCE, LONG_START_SEQUENCE}

A conforming compressed data must consist of only meaningful window_sequence transitions. However, decoders are required to handle non-meaningful window_sequence transitions as well. The performance requirements for non-meaningful window_sequence transitions are the same as for the meaningful transitions.

max_sfb: must be <= num_swb_long or num_swb_short as appropriate for window_sequence and sampling frequency.

ltp_data_present[ch] : must be set to zero.

7.6.1.2.2.4 extended_bsac_raw_data_block()

channel_configuration_index : shall be encoded with the value which is less than 6

The restrictions on the extended_bsac_raw_data_block() shall be applied as those of bsac_raw_data_block().

7.6.2 Decoders

7.6.2.1 Characteristics

A conforming decoder may also support any of the following modifications to the parameters in an audio compressed data:

Table 24 — BSAC Parameter

Compressed data Characteristic	Variation
sampling rate	a decoder may support additional sampling rates beyond the minimums listed for its profile and level
audio channels	a decoder may support additional channel elements beyond the minimums listed for its profile and level

7.6.2.2 Test procedure

To test audio decoders, ISO/IEC JTC 1/SC 29/WG 11 supplies a number of test sequences which are provided for sampling rates of 8, 11.025, 12, 16, 22.05, 24, 32, 44.1, 48 kHz. The test set includes a sine sweep and musical test sequences, as listed in Table 25. They cover the wide range of sampling rate, bit rate, number of channels and AAC related tools (window_shape, M/S stereo). The extension _fs is appended to the compressed data name to indicate the sampling rate of the test sequence. Possible values of fs are 8, 11, 12, 16, 22, 24, 32, 44 and 48 corresponding to the possibly non-integer sampling rates listed above. For each compressed data, two bitrates are listed in Table 25. The lower bitrate is to be used for sampling rates of 16kHz and below, and the higher bitrate is to be used at sampling rates above 16 kHz.

Fine grain scalability would create large overhead if one would try to transmit fine grain layers over multiple elementary streams (ES). So, in order to reduce overhead and implement the fine grain scalability efficiently in current MPEG-4 system, the server can organize the Access Unit (AU) by grouping the fine grain layres into the large-step layers. Then the AU is transmitted over ES. For each compressed data, the number of ES to be transmitted and the bitrates of each ES are listed in Table 25. In case of epConfig=1, the base layer is split into the BSAC common side information AU and the remaining base layer AU depending on the error categories. The lower bitrate is to be used for sampling rates of 16kHz and below, and the higher bitrate is to be used at sampling rates above 16 kHz.

In case of a BSAC compressed data whose top layer is *n*, downscaled audio representations are tested as a conformance point. The PCM output at the highest layer **highestLayer** of a decoder under test is compared with a reference output, where **highestLayer** is the highest layer of the scalable configuration used for

decoding (starting with 0 for the base layer). The highest layers used for the comformance testing are listed in Table 25. The lower **highestLayer** is to be used for sampling rates of 16 kHz and below, and the higher **highestLayer** is to be used at sampling rates above 16 kHz.

The following test procedure applies to all sine sweep signals: Testing is done by comparing the output of a decoder under test with a reference output also supplied by ISO/IEC JTC 1/SC 29/WG 11 using the procedure described in 7.1.2.2.1. This test only verifies the computational accuracy of an implementation.

For the remaining test sequences, a check of conformance using the LSB criterion or other measurements (e.g. objective perceptual measurement systems) is not mandatory, but highly recommended.

A conforming decoder shall support all characteristics given by the definition of level in the Mobile Audio Internetworking profile and the Natural Audio profile. Thus only compressed data belonging to the specific level & profile have to be tested.

The conformance bitstreams corresponding to BSAC Extensions for the multichannel extension are the “er_bs09” ~ “er_bs11” as listed in.Table 25 The conformance testing criterion for the multichannel extension is same as ER BSAC.
The conformance bitstreams corresponding to BSAC Extensions for the SBR extension are listed in Table 26. The conformance testing criterion for the SBR extension is specified in 7.17. The SBR conformance test method of BSAC Extensions is different from that of HE-AAC in terms of parsing the SBR payload. In Figure 11, the “Read store/input” module is to parse the SBR payload from BSAC Extensions payload which can be realized by either decoding the whole BSAC Extensions payload or seeking the “*sync_word*” in the extensions payload.
The “coreSetup” of naming convention for BSAC Extensions is the “xx_sbr_<nchan>_<fs>” where the “nchan” represents the number of channel and the “fs” is the sampling rate of SBR tool. By comparing the decoded wave with the reference wave for the different “highestLayer”, the “HF overlap” decoding module specified in 4.5.2.11.1.4, ISO/IEC 14496-3:2009 can be checked if it's implemented correctly. The testing criterion with “Diffmax” and “RMS max” shall be applied only when the bitstream is decoded up to top layer.

7.6.2.3 Test sequences

Table 25 — ER BSAC Object Type Test Compressed data for Mobile Audio Internetworking Profile Level 1-3 and Natural Audio Profile Level 1-2

File base name	Content	Base Layer Bitrate (kbit/s)	Top Bitrate (kbit/s)	Top Layer (n)	Number of ES	ES Bitrate (kbit/s)	Number of channel	Intensity	MS	TNS	PNS	epConfig	SBA	Highest Layer	Test Procedure
er_bs01_ep0	sine swee p	16	40/64	24/48	1	40/64	1					0		24/48	RMS
er_bs01_ep1	sine swee p	16	40/64	24/48	6	BL1, BL2,6/12, 6/12, 6/12, 6/12	1					1		24/48	RMS

File base name	Content	Base Layer Bitrate (kbit/s)	Top Bitrate (kbit/s)	Top Layer (n)	Number of ES	ES Bitrate (kbit/s)	Number of channel	Intensity	MS	TNS	PNS	epConfig	SBA	Highest Layer	Test Procedure
er_bs02_ ep0	music	16	40/64	24/48	25/49	BL,1, 1, ..., 1, 1	1					0		0, 1, 2, ... , 24/48	
er_bs02_ ep1	music	16	40/64	24/48	6	BL1, BL2,6 /12, 6/12, 6/12, 6/12	1					1		0, 1, 2, ... , 24/48	
er_bs03_ ep0	music	32	80/128	24/48	2	BL,24 /48	2					0		0, 24/48	
er_bs03_ ep1	music	32	80/128	24/48	6	BL1, BL2, 12/24,12/2 4,12/24,12 /24	2					1		0, 24/48	
er_bs04_ ep0	music	32	80/128	24/48	4	BL,24 (48), 12(24), 12(24)	2		Yes	Yes		0		0, 12/24, 18/36, 24/48	
er_bs04_ ep1	music	32	80/128	24/48	6	BL1, BL2, 12/24,12/2 4,12/24,12 /24	2		Yes	Yes		1		0, 12/24, 18/36, 24/48	
er_bs05_ ep0	music	32	80/128	24/48	5	BL,12/24, 12/24,12/2 4,12/24	2	Yes	Yes	Yes		0		0, 6/12, 12/24, 18/36, 24/48	
er_bs05_ ep1	music	32	80/128	24/48	6	BL1, BL2, 12/24,12/2 4,12/24,12 /24	2	Yes	Yes	Yes		1		0, 6/12, 12/24, 18/36, 24/48	
er_bs06_ ep0	music	32	80/128	24/48	3	BL, 24/48, 24/48	2	Yes	Yes	Yes		0	Yes	0, 12/24, 24/48	
er_bs06_ ep1	music	32	80/128	24/48	6	BL1, BL2, 12/24,12/2 4,12/24,12 /24	2	Yes	Yes	Yes		1	Yes	0, 6/12, 12/24, 18/36, 24/48	

File base name	Content	Base Layer Bitrate (kbit/s)	Top Bitrate (kbit/s)	Top Layer (n)	Number of ES	ES Bitrate (kbit/s)	Number of channel	Intensity	MS	TNS	PNS	epConfig	SBA	Highest Layer	Test Procedure
er_bs07_ ep0	noise	16	40/64	24/48	1	40/64	1				Yes	0		24/48	PNS-1
er_bs07_ ep1	noise	16	40/64	24/48	6	BL1, BL2, 6/12, 6/12, 6/12, 6/12	1				Yes	1		24/48	PNS-1
er_bs08_ ep0	noise	16	40/64	24/48	1	40/64	1				Yes	0		24/48	PNS-2/3
er_bs08_ ep1	noise	16	40/64	24/48	6	BL1, BL2, 6/12, 6/12, 6/12, 6/12	1				Yes	1		24/48	PNS-2/3
er_bs09_ ep0	music	32	200/320	24/48	1	200/320	5 {1,0,3} (**)		Yes	Yes		0		24/48	
er_bs09_ ep1	music	32	200/320	24/48	6	BL, 48/96, BL, 24/48, BL, 48/96	5 {1,0,3} (**)		Yes	Yes		1		0,24/48, 0,24/48, 0, 24/48 (*)	
er_bs10_ ep0	music	32	120/192	24/48	1	120/192	3 {1,0} (**)		Yes	Yes		0		24/48	
er_bs10_ ep1	music	32	120/192	24/48	4	BL, 48/96 ,BL, 24/48	3 {1,0} (**)		Yes	Yes		1		0, 24/48, 0, 24/48 (*)	
er_bs11_ ep0	music	32	300/480	24/48	1	300/480	8 {1,0,4, 3,5} (**)		Yes	Yes		0		24/48	
er_bs11_ ep1	music	32	300/480	24/48	9	BL, 48/96, BL, 24/48, ,20/32 ,BL, 48/96, BL, 48/96	8 {1,0,4, 3,5} (**)		Yes	Yes		1		0, 24/48, 0,24/48, 0,4/16, 0,24/48, 0,24/48 (*)	

where, ES : Elementary Stream

BL : Base Layer Bitrate

BL1 : Bitrate of the BSAC common side information

BL2 : Bitrate of Base Layer except the BSAC common side information.

* : The definition of 'highestLayer' for this bitstream is the highest layer of each BSAC extension payloads.

** : The numbers in the bracket represent the "channel_configuration_index" of each BSAC extension payload as defined in ISO/IEC 14496-3:2009, 4.5.2.6.2.1.1, Table 4.100 - channel_configuration_index.

Table 26 — ER BSAC Object Type with SBR tool

file base name	tool	content	bitrate (kbit/s)	nch	Core/SBR Sampling rate	QMF Identification	QMF Accuracy	Envelope Adjuster Accuracy	Grid control tests	Header Change Tests	Inverse Filtering Tests	Additional Sines Tests	CRC	Diff max	RMS max (linear value)	highestLayer	test procedure
er_bs12_sbr	twi	none	64	1	24/48kHz	y	y	-	-	-	-	y	-	-	-	0, 24, 48	
er_bs13_sbr	qmf	Sine Sweep	64	1	16/32kHz 22/44kHz 24/48kHz	-	Y	-	-	-	-	-	-	5	1.4	0, 24, 48	maxDiff/RMS
er_bs14_sbr	e	rectangle * 10Hz sine	128	2	16/32kHz 22/44kHz 24/48kHz	-	-	y	-	-	-	-	y[A]	90	2.0	0, 24, 48	maxDiff/RMS
er_bs15_sbr	gh	rectangle * 10Hz sine	64/128	1/2	16/32kHz 22/44kHz 24/48kHz	-	-	-	y	y	-	-	-	51	1.5	0, 24, 48	maxDiff/RMS
er_bs16_sbr	i	rectangle + noise	64/128	1/2	16/32kHz 22/44kHz 24/48kHz	-	-	-	-	-	y	-	y[A]	36	3.4	0, 24, 48	maxDiff/RMS
er_bs17_sbr	s	noise	64/128	1/2	16/32kHz 22/44kHz 24/48kHz	-	-	-	-	-	-	y	-	120	1.9	0, 24, 48	maxDiff/RMS
er_bs18_sbr	cm	music	64/128	1/2	16/32kHz 22/44kHz 24/48kHz	-	-	-	-	-	-	-	-	-	-	0, 24, 48	
er_bs19_sbr	cm	music	320	5.1	24/48kHz	-	-	-	-	-	-	-	-	-	-	24/48	
er_bs20_sbr	sig0	music	128/320	2/5	24/48kHz	-	-	-	-	-	-	-	-	-	-	24/48	
er_bs21_sbr	sig1	music	128/320	2/5	24/48kHz	-	-	-	-	-	-	-	-	-	-	24/48	
er_bs22_sbr	sig2	music	128/320	2/5	24/48kHz	-	-	-	-	-	-	-	-	-	-	24/48	
er_bs23_sbr	sr	music	128	2	16/32kHz 22/44kHz 24/48kHz 48/96kHz	-	-	-	-	-	-	-	-	-	-	24/48	

[A] **CRC enabled for 32 kHz testvectors**

7.7 CELP

7.7.1 DecoderSpecificInfo Characteristics

Bitstreams provided may apply restrictions to the following syntactic elements of the Object Descriptor Stream:

a) AudioObjectType
b) samplingFrequencyIndex
c) samplingFrequency
d) channelConfiguration
e) SampleRateMode
f) RPE _Configuration
g) MPE _Configuration
h) NumEnhLayers
i) BandwidthScalabilityMode
j) isBaseLayer
k) isBWSLayer
l) CELP-BRS-id

7.7.2 Audio Access Unit characteristics

Bitstream providers may apply restrictions to the following syntactic elements of the bitstream:

m) LPC_Present
n) interpolation_flag
o) gain_indices [1]

7.7.3 Procedure to Test Bitstream Conformance

In case that DecoderConfigDescriptor() (see ISO/IEC 14496-1 MPEG4 Systems) is used for MPEG-4 CELP audio decoders, the Audio DecoderSpecificInfo must comply with the semantic conditions described below:

AudioSpecificConfig - Scalable or Main Profile

When the CELP object type is used as part of the Scalable Profile or the Main Profile, the following restrictions apply to the AudioSpecificConfig:

AudioObjectType: must be set to 8 for CELP object types.

channelConfiguration: must be set to 1.

AudioSpecificConfig – Speech Profile

When the CELP object type is used as part of the Speech Profile, the following restrictions apply to the AudioSpecificConfig:

AudioObjectType: must be set to 8 for CELP object types.

samplingFrequencyIndex: must be set to 0xb or 0x8.

channelConfiguration: must be set to 1.

CELP bitstreams must comply with the semantic conditions described below.

CelpHeader – Scalable or Main Profile

When the CELP object type is used as part of the Scalable Profile or the Main Profile, the following restrictions apply to the CelpHeader fields:

SampleRateMode: When the CELP object type is used as a core codec in a CELP/AAC scalable bitstream, the SampleRateMode field must equal 8KHZ.

ExcitationMode: When SampleRateMode equals 8KHZ, the ExcitationMode field must equal MPE.

RPE_Configuration: this unsigned integer element shall not exceed 3.

MPE_Configuration: when the SampleRateMode field equals 8KHZ, the unsigned integer element shall not exceed 27. When the SampleRateMode field equals 16KHZ, this element shall not be encoded with 7 or 23.

NumEnhLayers: when MPE_Configuration equals 27 and SampleRateMode equals 8KHZ, this field must equal 0.

BandwidthScalabilityMode: this field must equal OFF when SampleRateMode equals 16KHZ. When MPE_Configuration equals 27 and SampleRateMode equals 8KHZ, this field must equal OFF.

CelpHeader – Speech Profile

When the CELP object type is used as part of the Speech Profile, the following restrictions apply to the CelpHeader fields:

SampleRateMode: in case DecoderConfigDescriptor() is used, the SampleRateMode field must equal 8KHZ when samplingFrequencyIndex equals 0xb. This field must equal 16KHZ when samplingFrequencyIndex equals 0x8.

ExcitationMode: when SampleRateMode equals 8KHZ, the ExcitationMode field must equal MPE.

RPE_Configuration: this unsigned integer element shall not exceed 3.

MPE_Configuration: when the SampleRateMode field equals 8KHZ, the unsigned integer element shall not exceed 27. When the SampleRateMode field equals 16KHZ, this element shall not be encoded with 7 or 23.

NumEnhLayers: when MPE_Configuration equals 27 and SampleRateMode equals 8KHZ, this field must be 0.

BandwidthScalabilityMode: this field must equal OFF when SampleRateMode equals 16KHZ. When MPE_Configuration equals 27 and SampleRateMode equals 8KHZ, this field must equal OFF.

Celp_LPC

LPC_Present: when FineRateControl equals ON and interpolation_flag equals 1, this bit shall not be set to “0”. In the first frame in a CELP bitstream, directly following the CelpHeader, this field shall be set to “1”. If frame number n in a bitstream has LPC_Present set to “0”, frame n+1 shall have LPC_Present set to “1”.

interpolation_flag: if frame number n in a bitstream has LPC_Present set to “1” and interpolation_flag set to “1”, frame n+1 shall have interpolation_flag set to “0”.

RPE_frame

gain_indices [1]: for subframe 0 in every RPE_frame, this unsigned integer element shall not be encoded with 31.

isBaseLayer: shall be set to 1 when the audio data of the base layer is transmitted, and shall be set to 0 when the audio data of the enhancement layer is transmitted.

isBWSLayer: shall be set to 1 when the audio data of the bandwidth scalable enhancement layer is transmitted, and shall be set to 0 when the audio data of the bit-rate scalable enhancement layer is transmitted.

CELP-BRS-id: shall not be set to 0.

7.7.4 Decoder Characteristics

Main Profile

When the CELP decoder is used as a part of the Main Profile, the decoder must meet the level requirements as described in ISO/IEC 14496-3, subpart 1. Note that in case of a scalable decoder, the level complexity boundaries are applicable to the entire decoder. No complexity bounds are defined for the CELP object type decoder separately.

Scalable Profile

When the CELP decoder is used as a part of the Scalable Profile, the decoder must meet the level requirements as described in ISO/IEC 14496-3, subpart 1. Note that in case of a scalable decoder, the level complexity boundaries for level 4 are applicable to the entire decoder. No complexity bounds are defined for the CELP object type decoder separately.

Speech Profile

When the CELP decoder is used as a part of the Speech Profile, a conforming decoder must support a minimum number of Audio object types in the Speech profile. For level 1 in the Speech profile, a decoder has to support at least one audio object. For level 2 in the Speech profile, a decoder has to support at least 20 audio objects simultaneously.

7.7.5 Procedure to Test Decoder Conformance

To test audio decoders, the electronic attachment to this part of ISO/IEC 14496 supplies a number of test sequences. Supplied sequences cover CELP decoders and are provided for sampling rates of 8 and 16 kHz. The test set covers an orthogonal subset of all MPEG-4 CELP modes. This test only verifies the functionality and the computational accuracy of a CELP decoder implementation.

For a supplied test sequence, testing can be done by comparing the output of a decoder under test with a reference output, also supplied by the electronic attachment to this part of ISO/IEC 14496. Any post-processing and pre-pitch filtering available in the decoder under test and in the Reference decoder must be disabled while compliance is tested. Measurements are carried out relative to full scale where the output signals of the decoders are normalized to be in the range between -1 and +1.

Two levels of accuracy are defined for the CELP decoder conformance testing procedure.

Full Accuracy: A decoder meeting the Full Accuracy conformance requirements as defined below may be called a Full Accuracy conformant decoder. This level of accuracy is intended for CELP decoders running on floating-point platforms.

Fixed-Point Accuracy: A decoder may be called conformant with Fixed-Point Accuracy in case the Fixed-Point Accuracy conformance criteria are met, as defined below. This level of accuracy is targeted at CELP decoders with a limited accuracy due to fixed-point internal calculations.

Conformance criterion for Full Accuracy CELP decoders

The RMS/LSB Measurement test procedure applies (see 7.1.2.2.1)

Conformance criteria for Fixed-Point Accuracy CELP decoders

The conformance criteria for Fixed-Point Accuracy decoders are based on measuring the segmental SNR and the LPC cepstral distortion (CD) between the Reference decoder output and the output of the decoder to be tested. The segment length to be used in the calculation of the SNR and CD is equal to the CELP frame length. The SNR and the CD have to be calculated only for the segments of which the power of the Reference signal is in the range [-50...-15] dB. CD is defined as

$$CD = \frac{10}{\ln(10)} \cdot \sqrt{2D}$$

D is the accumulated distortion of the LPC cepstrum C_{ref} of the reference signal and C_{test} of the output of the decoder under test. D is defined as

$$D = \sum_{i=1}^{N} \left(C_{ref}[i] - C_{test}[i]\right)^2$$

N is the LPC cepstrum order which equals 32. The LPC cepstrum C[i] is defined by means of the algorithm `lpc2cepstrum` based on the LPC coefficients of a 16th order linear prediction filter. The computation of the LPC filter coefficients lpc_coef [j] is defined by the algorithm `calculate_lpc`.

To be called an ISO/IEC 14496-3 CELP decoder with Fixed-Point Accuracy

the average value of the segmental SNR shall exceed 30 dB

and in addition,

the average value of the CD shall not exceed 1 dB.

7.7.6 Descriptions of the audio test bitstreams

Table 27 — Test Bitstreams for the CELP object type: MPE modes

File Name	CE00	CE01	CE02	CE03	CE04	CE05	CE06
Bitrate [bps]	8300	17900	4250+ 3x2000	16000 + 2x4000	12000	5200 + 10667	6000 + 2000 + 12400
Sampling rate [kHz]	8	16	8	16	8	8/16	8/16
Excitation mode	MPE	MPE	MPE	MPE	MPE	MPE	MPE
MPE_Configuration	14	11	1	20	25	4	7
FineRate control	No	No	No	No	Yes	No	No
Bitrate scalability	No	No	Yes	Yes	No	No	Yes
NumEnhLayers	0	0	3	2	0	0	1
Bandwidth scalability	No	No	No	No	No	Yes	Yes
BWS_Configuration	n.a.	n.a.	n.a.	n.a.	n.a.	1	2

Table 28 — Test Bitstreams for the CELP object type: RPE modes

File Name	CE07	CE08	CE09	CE10	CE11	CE12	CE13	CE14
Bitrate [bps]	14400	14000	16000	16000	18667	18000	22533	22000
Sampling rate [kHz]	16	16	16	16	16	16	16	16
Excitation mode	RPE	RPE	RPE	RPE	RPE	RPE	RPE	RPE
RPE_Configuration	0	0	1	1	2	2	3	3
FineRate control	No	Yes	No	Yes	No	Yes	No	Yes
Bitrate scalability	No	No	No	No	No	No	No	No
Bandwidth scalability	No	No	No	No	No	No	No	No

7.8 ER CELP

The ER CELP object is an extension of the CELP object and supports silence compression and error resilience functionalities. The silence compression significantly reduces the average transmission bitrate of silent input signals.

7.8.1 Compressed data

7.8.1.1 Characteristics

7.8.1.1.1 AudioSpecificConfig

Compressed data provided may apply restrictions to the following syntactic elements of the Object Descriptor Stream:

AudioObjectType

samplingFrequencyIndex

samplingFrequency

channelConfiguration

SampleRateMode

RPE _Configuration

MPE _Configuration

NumEnhLayers

BandwidthScalabilityMode

7.8.1.1.2 Bitstream payload

Compressed data providers may apply restrictions to the following syntactic elements of the compressed data:

LPC_Present

interpolation_flag

gain_indices[1]

7.8.1.2 Test procedure

In case that DecoderConfigDescriptor() (see ISO/IEC 14496-1 MPEG-4 Systems) is used for MPEG-4 V2 CELP audio decoders, the AudioSpecificConfig must comply with the semantic conditions described below:

7.8.1.2.1 AudioSpecificConfig

The following restrictions apply to the AudioSpecificConfig:

AudioObjectType: This field must be set to 24 for ER CELP objects.

samplingFrequencyIndex: This field must be set to 0xb or 0x8.

channelConfiguration: This field must be set to 1.

ER CELP compressed data must comply with the semantic conditions described below.

7.8.1.2.2 ER_SC_CelpHeader

The following restrictions apply to the ER_SC_CelpHeader fields:

SampleRateMode: In case DecoderConfigDescriptor() is used, the SampleRateMode field must equal 8KHZ when samplingFrequencyIndex equals 0xb. This field must equal 16KHZ when samplingFrequencyIndex equals 0x8.

ExcitationMode: When SampleRateMode equals 8KHZ, the ExcitationMode field must equal MPE.

RPE_Configuration: This unsigned integer element shall not exceed 3.

MPE_Configuration: When the SampleRateMode field equals 8KHZ, the unsigned integer element shall not exceed 27. When the SampleRateMode field equals 16KHZ, this element shall not be encoded with 7 or 23.

NumEnhLayers: When MPE_Configuration equals 27 and SampleRateMode equals 8KHZ, this field must be 0.

BandwidthScalabilityMode: This field must equal OFF when SampleRateMode equals 16KHZ. When MPE_Configuration equals 27 and SampleRateMode equals 8KHZ, this field must equal OFF.

7.8.1.2.3 Celp_LPC

LPC_Present: When FineRateControl equals ON and interpolation_flag equals 1, this bit shall not be set to “0”. In the first frame in an ER CELP compressed data, directly following the ER_SC_CelpHeader, this field shall be set to “1”. If frame number *n* in a compressed data has LPC_Present set to “0”, frame *n*+1 shall have LPC_Present set to “1”. When Fine Rate Control equals ON and SilenceCompression equals ON, in the first active frame (TX_flag=1) following a non-active frame (TX_flag=0, 2 or 3), this field shall be set to “1”.

interpolation_flag: If frame number *n* in a compressed data has LPC_Present set to ‘1’ and interpolation_flag set to “1”, frame *n*+1 shall have interpolation_flag set to “0”.

7.8.1.2.4 RPE_frame

gain_indices [1]: For subframe 0 in every RPE_frame, this unsigned integer element shall not be encoded with 31.

7.8.2 Decoders

7.8.2.1 Characteristics

7.8.2.1.1 High Quality Audio Profile

When the ER CELP decoder is used in the High Quality Audio Profile, the decoder shall meet the level requirements as described in ISO/IEC 14496-3. The decoder shall support at least one audio object with one channel for Level 1. No complexity bounds are separately defined for the ER CELP object decoder.

7.8.2.1.2 Low Delay Audio Profile

When the ER CELP decoder is used in the Scalable Profile, the decoder shall meet the level requirements as described in ISO/IEC 14496-3 . The decoder shall support one audio object for Levels 1 and 2. No complexity bounds are separately defined for the ER CELP object decoder.

7.8.2.1.3 Natural Audio Profile

When the ER CELP decoder is used in the Main Profile, the decoder shall meet the level requirements as described in ISO/IEC 14496-3. Note that in case of a scalable decoder, the level complexity boundaries are applicable to the entire decoder. No complexity bounds are separately defined for the ER CELP object decoder.

7.8.2.2 Test procedure

To test audio decoders, ISO/IEC JTC 1/SC 29/WG 11 supplies a number of test sequences. Supplied sequences cover ER CELP decoders and are provided for sampling rates of 8 and 16 kHz. The test set covers an orthogonal subset of all MPEG-4 ER CELP modes. This test only verifies the functionality and the computational accuracy of a V2 CELP decoder implementation.

For a supplied test sequence, testing can be done by comparing the output of a decoder under test with a reference waveform, also supplied by ISO/IEC JTC 1/SC 29/WG 11. Any post-processing and pre-pitch filtering available in the decoder under test and in the Reference decoder must be disabled while compliance is tested. Measurements are carried out relative to full scale where the output signals of the decoders are normalized to be in the range between -1 and +1.

Two levels of accuracy are defined for the ER CELP decoder conformance testing procedure:

Table 29

Full Accuracy	A decoder meeting the Full Accuracy conformance requirements as defined below may be called a Full Accuracy conforming decoder. This level of accuracy is intended for ER CELP decoders running on floating-point platforms.
Fixed-Point Accuracy	A decoder may be called conforming with Fixed-Point Accuracy in case the Fixed-Point Accuracy conformance criteria are met, as defined below. This level of accuracy is targeted at ER CELP decoders with a limited accuracy due to fixed-point internal calculations.

Conformance criteria for Full Accuracy ER CELP decoders

A Full Accuracy ER CELP decoder at an accuracy level of “K bit” has to fulfill the RMS/LSB criterion as defined in 7.1.2.2.1.

Conformance criteria for Fixed-Point Accuracy CELP decoders

The conformance criteria for Fixed-Point Accuracy decoders are based on measuring the segmental SNR and the LPC cepstral distortion (CD) between the reference waveform and the output of the decoder to be tested. The segment length to be used in the calculation of the SNR is equal to the ER CELP frame length. The SNR has to be calculated only for the segments of which the power of the reference waveform is in the range [-50...-15] dB. CD is defined as

$$CD = \frac{10}{\ln(10)} \cdot \sqrt{2D}$$

D is the accumulated distortion of the LPC cepstrum Cref of the reference waveform and Ctest of the output of the decoder under test. D is defined as

$$D = \sum_{i=1}^{N} \left(C_{ref}[i] - C_{test}[i] \right)^2$$

N is the LPC cepstrum order that equals 32. The LPC cepstrum C[i] is defined by means of the algorithm lpc2cepstrum based on the LPC coefficients of a 16th order linear prediction filter. The computation of the LPC filter coefficients lpc_coef [j] is defined by the algorithm calculate_lpc.

To be called an ISO/IEC 14496-3 ER CELP decoder with Fixed-Point Accuracy

the average value of the segmental SNR shall exceed 30 dB and in addition, the average value of the CD shall not exceed 1 dB.

7.8.2.3 Test sequences

Table 30 — Test sequences for the ER CELP object type: MPE modes

File base name	er_ce00_ep0, 1	er_ce01_ep0, 1	er_ce02_ep0, 1	er_ce03_ep0, 1	er_ce04_ep0, 1
Reference signal	er_ce00	er_ce01	er_ce02	er_ce03	er_ce04
Nominal bitrate [bps]	3900	4967	6000	8400	11200
Sampling rate [kHz]	8	8	8	8	8
Excitation mode	MPE	MPE	MPE	MPE	MPE
SilenceCompression	ON	ON	OFF	ON	ON
MPE_Configuration	0	3	7	14	22
FineRate control	No	No	No	No	No
Bitrate scalability	No	No	No	No	No
NumEnhLayers	0	0	0	0	0
Bandwidth scalability	No	No	No	No	No
BWS_Configuration	n.a.	n.a.	n.a.	n.a.	n.a.
epConfig	0, 1	0, 1	0, 1	0, 1	0, 1

File base name	er_ce05_ep0, 1	er_ce06_ep0, 1	er_ce07_ep0, 1	er_ce08_ep0, 1	er_ce09_ep0, 1
Reference signal	er_ce05	er_ce06_lay0 er_ce06_lay1 er_ce06_lay2 er_ce06_lay3	er_ce07	er_ce08	er_ce09
Nominal bitrate [bps]	12200	5267 + 2000 + 2000 + 10667	18000	13800	16000
Sampling rate [kHz]	8	8/16	16	16	16
Excitation mode	MPE	MPE	MPE	MPE	MPE
SilenceCompression	ON	ON	ON	ON	ON
MPE_Configuration	25	4	11	16	21
FineRate control	Yes	No	No	No	Yes
Bitrate scalability	No	Yes	No	No	No
NumEnhLayers	0	2	0	0	0
Bandwidth scalability	No	Yes	No	No	No
BWS_Configuration	n.a.	1	n.a.	n.a.	n.a.
epConfig	0, 1	0, 1	0, 1	0, 1	0, 1

The nominal bit rate represents the bit rate for the active frames (TX_flag = 1).

Table 31 — Test sequences for the ER CELP object type: RPE modes

File base name	er_ce10_ep0, 1	er_ce11_ep0, 1	er_ce12_ep0, 1	er_ce13_ep0, 1	er_ce14_ep0, 1
Reference signal	er_ce10	er_ce11	er_ce12	er_ce13	er_ce14
Nominal bitrate [bps]	14533	14000	16200	16000	16000
Sampling rate [kHz]	16	16	16	16	16
Excitation mode	RPE	RPE	RPE	RPE	RPE
SilenceCompression	ON	ON	ON	OFF	ON
RPE_Configuration	0	0	1	1	1
FineRate control	No	Yes	No	Yes	Yes
Bitrate scalability	No	No	No	No	No
Bandwidth scalability	No	No	No	No	No
epConfig	0, 1	0, 1	0, 1	0, 1	0, 1

File base name	er_ce15_ep0,1	er_ce16_ep0,1	er_ce17_ep0,1	er_ce18_ep0,1
Reference signal	er_ce15	er_ce16	er_ce17	er_ce18
Nominal bitrate [bps]	18800	18000	22667	22000
Sampling rate [kHz]	16	16	16	16
Excitation mode	RPE	RPE	RPE	RPE
SilenceCompression	ON	ON	ON	ON
RPE_Configuration	2	2	3	3
FineRate control	No	Yes	No	Yes
Bitrate scalability	No	No	No	No
Bandwidth scalability	No	No	No	No
epConfig	0, 1	0, 1	0, 1	0, 1

The nominal bit rate represents the bit rate for the active frames (TX_flag = 1).

7.9 HVXC

The HVXC object type is supported by the parametric speech coding (HVXC) tools, which provide fixed bitrate modes (2.0-4.0kbit/s) in a scalable and a non-scalable scheme, a variable bitrate mode (< 2.0kbit/s) and the functionalities of pitch and speed change. Only 8 kHz sampling rate and mono audio channel are supported.

7.9.1 DecoderSpecificInfo Characteristics

Bitstream provider must apply restrictions to the following parameters of the DecoderSpecificInfo:

a) AudioObjectType
b) samplingFrequencyIndex
c) channelConfiguration
d) HVXCrateMode
e) HVXCvarMode
f) isBaseLayer

7.9.2 Audio Access Unit Characteristics

Bitstream provider may apply no restrictions to any parameters of the bitstream.

7.9.3 AudioSource Fields Information Characteristics

A conforming decoder may support any of the following modification of some parameters:

a) speed change factor: A possible variation is from 0.5 to 2.0 (defined as spd in ISO/IEC 14496-3:2009, subpart 2, 2.5.5).
b) pitch change factor: A possible variation is from 0.5 to 2.0 (defined as pch_mod in ISO/IEC 14496-3:2009, subpart 2, 2.5.3).
c) test_mode: An interface to control some elements which are generated by random number generators. Its configuration is described below. This control interface is only for decoder conformance testing.

Table 32 — Description of test_mode

test_mode	Description
0000 0000 0000 0000	Normal operation mode as described in the standard
xxxx xxxx xxxx xxx1	post filter and post processing are skipped
xxxx xxxx xxxx xx1x	Initial values of harmonic phase are reset to zeros in Voiced Component Synthesizer
xxxx xxxx xxxx x1xx	Noise component addition is disabled in Voiced Component Synthesizer
xxxx xxxx xxxx 1xxx	Noise component generation is disabled in speed change mode and variable rate mode.
xxxx xxxx xxx1 xxxx	Reserved

(x: don't care)

It is recommended to have a "Private Test Information" input to set the **test_mode** to perform conformance testing thoroughly. If the decoder does not have such a control interface, limited procedures could be applied.

7.9.4 Procedure to Test Bitstream Conformance

7.9.4.1 DecoderSpecificInfo

The Audio must comply with the semantic conditions described below.

AudioObjectType: must be set to 9 (HVXC object type).

SamplingFrequencyIndex: must be set to 0xb (8000Hz).

ChannelConfiguration: must be set to 1.

HVXCrateMode: 2 bit identifier, which configures the bitrate of HVXC Object type must not exceed 2. When HVXCvarMode is set to 1(variable rate), HVXCrateMode must be set to 0 (2kbps).

7.9.4.2 Audio Access Unit.

No restrictions to the Audio Access Unit.

isBaseLayer: shall be set to 1 when the audio data of the base layer is transmitted, and shall be set to 0 when the audio data of the enhancement layer is transmitted.

7.9.5 Decoder Characteristics

A conforming decoder may support any of the following modifications of some parameters in audio bitstreams.

a) bitrate

b) variable rate(fixed rate/variable rate)

A conforming decoder shall support one or both of the delay mode (normal delay mode/low delay mode), where the delay mode does not exist in audio bitstreams.

7.9.6 Procedure to Test Decoder Conformance

HVXC decoder uses independent random number generators for

- initial values of harmonic phase in Voiced Component Synthesizer
- noise component addition in Voiced Component Synthesizer
- noise components in speed change decode
- noise components in variable rate mode decode

For that reason, decoder conformance can not be tested by direct comparison and specific testing procedures are necessary.

In this Subclause, the following testing procedures are described:

1. Procedures without a control interface

   1.1. All Voiced Bitstream

   1.2. All Unvoiced Bitstream (direct comparison by measuring segmental SNR)

2. Procedures with a control interface

   2.1. Direct Comparison by measuring segmental SNR (with random number generators disabled)

   2.2. Harmonic Phase Initialization (verification of phase randomness)

Any post-filtering and post-processing in the decoder under test must be disabled in testing decoder conformance because conformance point is placed before the informative post-filter and post-processing.

It should be noted that transition from “Voiced” to “Unvoiced” or from “Unvoiced” to “Voiced” can not be tested by “Procedures without a control interface”. To test decoder conformance thoroughly, it is recommended to have a control interface and to take “Procedures with a control interface” furthermore.

The software for calculating the comformance criteria is available together with the bitstreams.

Figure 3 shows the decoder output signal timing for testing decoder conformamce.

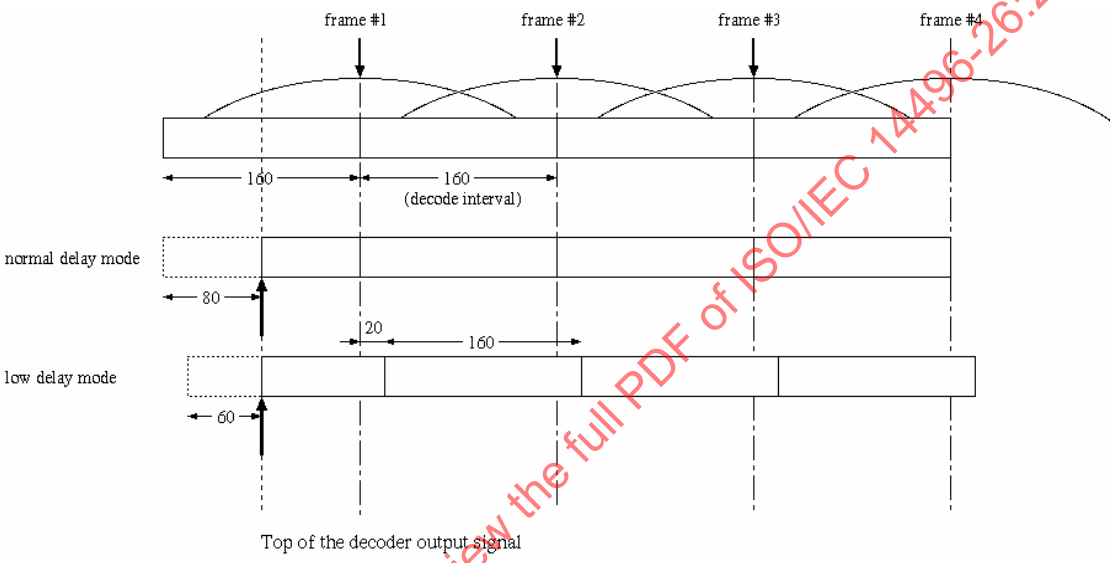

Figure 3 — Decoder output signal timing for testing decoder conformance

7.9.6.1 Procedures without a control interface

For the decoder which does not have a control interface to disable random number generators, the specialized test bitstreams are used:

- All Voiced bitstream (HV01 and HV02)
- All Unvoiced bitstream(HV03 and HV04)

For the former bitstream specialized testing procedure is applied. For the latter bitstream output signal is produced in deterministic way and direct comparison by measuring segmental SNR with reference signal is executed.

7.9.6.1.1 Procedures by All Voiced bitstream

In this procedure, the following specialized bitstreams (HV01 and HV02) are supplied:

- All of frames are “Voiced”.
- Pitch lag sweeping from 30 to 40 cyclically.
- LSP indices to provide almost “flat” response.
- A fixed set of indices of the spectral envelope shape and gain.

It should be noted that since harmonic phase initialization using random number generator occurs at the "Voiced" frame after two successive "Unvoiced" frames, for this "All Voiced" bitstream, harmonic phase initialization never occurs and "initial" phase values (all zeros) are used in harmonic synthesis.

This implies that for this test bitstream the output signal of decoder is produced in deterministic way except noise component addition in Voiced Component Synthesizer.

Testing Procedure:

1. Both the output signal of a decoder under test and the reference output signal (without noise component addition) are normalized to be in the range between –1.0 and +1.0.
2. For each normalized signal, 256pt. Hanning windowing and 256pt.FFT are executed. Definition of Hanning window:

$$hann(i) = \frac{1}{2}\left(1.0 - \cos\left(\frac{2\pi\, i}{255}\right)\right) \qquad (0 \le i \le 255)$$

3. The differential spectrum is calculated.
4. For obtained differential spectrum, 7-taps average filtering is executed to obtain smoother spectrum.
5. If all of amplitudes of the spectrum are within a certain range, it can be said that the decoder under the test satisfies the conformance condition.

For each test bitstream, HV01 and HV02, an acceptable range of differential spectrum is shown Figure 1 and Figure 2 respectively.

The output signals to be tested are the followings:

1) 2.0kbps fixed rate mode decode (HV01)
   a) normal speed/pitch decode
   b) pitch change decode (pitch change factors to be tested are 1.6 and 0.8)
   c) speed change decode (speed change factors to be tested are 1.5 and 0.75)
2) 4.0kbps fixed rate mode decode (HV02)
   a) normal speed/pitch change decode

The above testing procedure is executed using dedicated software provided by the electronic attachment to this part of ISO/IEC 14496.

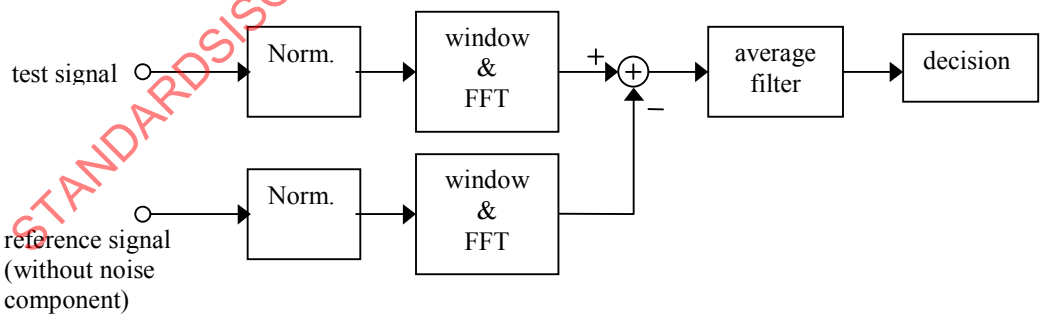

Figure 4 — Block Diagram of the Conformance Testing Procedure (All Voiced bitstream)

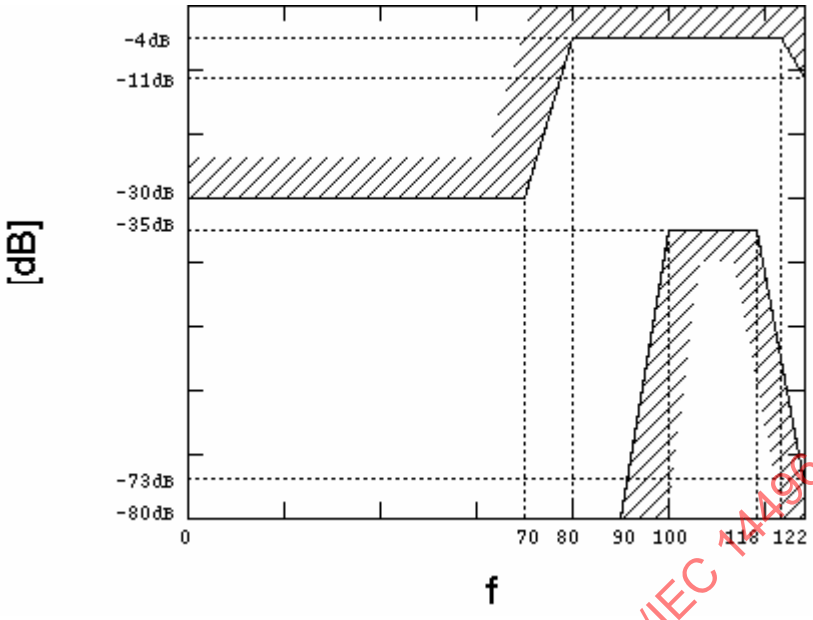

Figure 5 — Acceptable range of differential spectrum (for bitstream HV01)

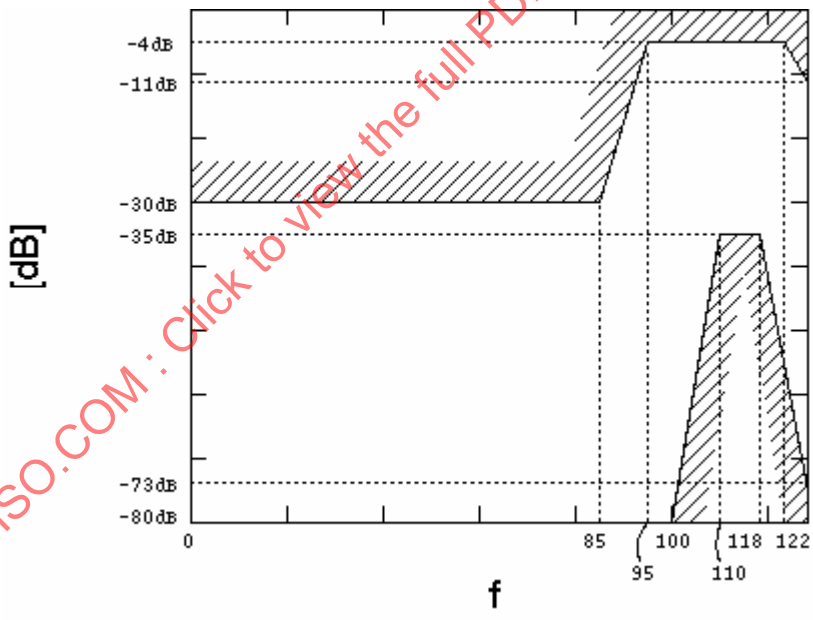

Figure 6 — Acceptable range of differential spectrum (for bitstream HV02)

Test Procedure Description using Pseudo C-code

```
#define NF 256 /* FFT length */
#define NI 160 /* frame interval */

void average_filter(
double *in,  /* in: input array */
double *out, /* out: output array */
int size,    /* in: size of average filter */
int tap      /* in: tap number of average filter(odd number) */
)
{


  for (i=0; i<tap/2; i++) {
    s[i]=0.0;
  }
  for (; i<size+tap/2; i++) {
    s[i]=in[i-tap/2];
  }
  for (; i<size+tap-1; i++) {
    s[i]=0.0;
  }

  for (i=0; i<size; i++) {
    out[i]=0.0;
    for (j=0; j<tap; j++) {
      out[i] += s[i+j-tap/2];
    }
    out[i] /= (double)tap;
  }
}

const double hann[NF];    /* Hanning window */
const double maxR[NF/2];  /* upper bound of acceptable range */
const double minR[NF/2];  /* lower bound of acceptable range */

void main()
{
  int i,k;
  double xa[..];    /* reference signal(normalized between -1.0 and 1.0) */
  double xb[..];    /* test signal(normalized between -1.0 and 1.0) */
  double re_a[NF],im_a[NF],re_b[NF],im_b[NF]; /* arrays for FFT */
  double ra[NF/2],rb[NF/2]; /* spectrum arrays */
  double dif[NF/2];  /* differential spectrum array */


  for (k=0; k<.. ; k++) {
    for (i=0; i<NF; i++) {
      re_a[i]=hann[i]*xa[NI*k+i];
      im_a[i]=0.0;
      re_b[i]=hann[i]*xb[NI*k+i];
      im_b[i]=0.0;
    }

    fft(re_a, im_a, 8); /* 256pt.FFT */
```

```
    fft(re_b, im_b, 8); /* 256pt.FFT */

    for (i=0; i<NF/2; i++) {
      ra[i]=sqrt(re_a[i]*re_a[i]+im_a[i]*im_a[i]);
      rb[i]=sqrt(re_b[i]*re_b[i]+im_b[i]*im_b[i]);
      dif[i]=fabs(ra[i]-rb[i]);
    }

    average_filter(dif, dif, NF/2, 7);

    for (i=0; i<NF/2; i++) {
      if (dif[i]>maxR[i] || dif[i]<minR[i]) {
        printf("conformance condition is not satisfied.\n");
      }
    }
  }
```

7.9.6.1.2 Procedures by All Unvoiced bitstream

Using supplied test bitstreams (HV03 and HV04), testing can be done by measuring segmantal SNR between the output signal of a decoder under test and a reference output signal.

To be called an ISO/IEC 14496-3 HVXC decoder, the segmental SNR defined in 7.1.2.2.2, must exceed 30[dB] (L=160).

The output signals to be tested are the followings:

1) 2.0kbps fixed rate mode decode (HV03)
   a) normal speed/pitch decode
2) 4.0kbps fixed rate mode decode (HV04)
   a) normal speed/pitch decode

7.9.6.2 Procedures with a control interface

For the decoder which have a control interface to disable random number generators, the following procedures are applied:

- Direct comparison by measuring segmental SNR (defined in 7.1.2.2.2) with random number generators disabled.
- Harmonic phase initialization (verification of phase randomness)

7.9.6.2.1 Direct Comparison with random number generators disabled

Using supplied test bitstreams (HV05, HV06 and HV07), output signal of a decoder under the test is produced with the following elements generated by random number generator disabled.

- Initial values of harmonic phase are reset to zeros in Voiced Component Synthesizer.
- noise component addition is disabled in Voiced Component Synthesizer
- noise components is disabled in speed change decode
- noise components is disabled in variable rate mode decode

Testing can be done by measuring segmental SNR between the output signal of a decoder under test and a reference output signal.

To be called an ISO/IEC 14496-3 HVXC decoder, the segmental SNR (defined in 7.1.2.2.2) must exceed 30[dB] (L=160).

The output signals to be tested are the followings:

1) 2.0kbps fixed rate mode decode (HV05)
   a) normal speed/pitch decode
   b) pitch change decode (pitch change factors to be tested are 1.6 and 0.8)
   c) speed change decode (pitch change factor to be tested are 1.5 and 0.75)
2) 4.0kbps fixed rate mode decode (HV06)
   a) normal speed/pitch change decode
3) variable rate mode decode(HV07)
   a) normal speed/pitch change decode

7.9.6.2.2 Harmonic Phase Initialization

In “Harmonic Excitation Synthesis” process of “Voiced Component Synthesizer”, harmonic phase values are initialized using random phase values uniformly distributing between 0 and 0.5π (see ISO/IEC 14496-3:2009, subpart 2, 2.5.6.3.2). In this Subclause, the specific testing procedure is presented to verify randomness of initial phases by measuring statistics of “peak/rms” over one pitch period only for “Voiced” frame.

For this procedure, specialised conformance bistream (HV08) are provided where

- V/UV decision is repeated every two frames(phase initialization occurs at the “Voiced” frame after two successive “Unvoiced” frames)
- for Voiced frame, pitch lag is 80 samples
- a fixed set of LSP indices to provide almost “flat” response (index of VQ without interframe prediction)
- a fixed set of indices of the spectral envelope shape and gain

A decoder under the test must produce output signal without noise component addition in Voiced Component Synthesizer (**test_mode** = xxxx xxxx xxxx x1x1).

Testing Procedure

1. A segment of one pitch period (80 samples) is taken as shown in the timing Figure 7. For successive two “Voiced” frames, three segments are obtained.
2. For each segment, peak is searched, rms value and “peak/rms” value are computed.
3. 1. and 2. are repeated for a whole decoder output signal.
4. Average and deviation of “peak/rms” value are computed.

If the obtained average and deviation of “peak/rms” is within a range shown in Table 33, it can be said that the decoder under the test satisfies the conformance condition.

The above testing procedure is executed using dedicated software.

Table 33 — Acceptable range of average and deviation of “peak/rms” value

test bitstream	HV08
bitrate[kbit/s]	2
Average	[5.68, 5.88]
Deviation	[0.32, 0.50]

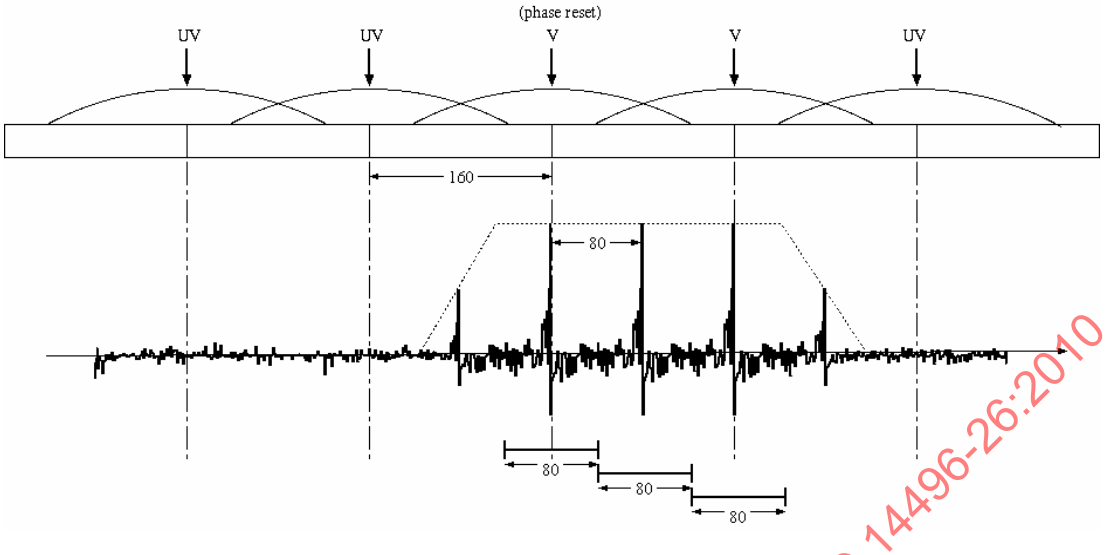

Figure 7 — An example decoder output signal and HVXC framing timing

7.9.7 Descriptions of the audio test bitstreams

Table 34 — HVXC object type test bitstream

File Name	HV01	HV02	HV03	HV04
Content	All Voiced	All Voiced	All Unvoiced	All Unvoiced
Sampling rate[kHz]	8	8	8	8
Bitrate[kbit/s]	2	4	2	4
Variable rate	No	No	No	No

Table 35 — HVXC object type test bitstream (continued)

File Name	HV05	HV06	HV07	HV08
Content				
Sampling rate[kHz]	8	8	8	8
Bitrate[kbit/s]	2	4	-	2
Variable rate	No	No	Yes	No

7.9.8 Descriptions of the audio reference output signal

Table 36 — HVXC object type reference output signal

File Name	HV01ref1	HV01ref2	HV01ref3	HV01ref4	HV01ref5
test bitstream	HV01	HV01	HV01	HV01	HV01
Sampling rate[kHz]	8	8	8	8	8
Bitrate[kbit/s]	2	2	2	2	2
Variable rate	No	No	No	No	No
Delay mode	normal	normal	normal	normal	normal
Pitch_change_factor	1	1.6	0.8	1	1
Speed_change_factor	1	1	1	1.5	0.75

Table 37 — HVXC object type reference output signal (continued)

File Name	HV01ref6	HV01ref7	HV01ref8	HV01ref9	HV01ref10
test bitstream	HV01	HV01	HV01	HV01	HV01
Sampling rate[kHz]	8	8	8	8	8
Bitrate[kbit/s]	2	2	2	2	2
Variable rate	No	No	No	No	No
Delay mode	low	low	low	low	low
Pitch_change_factor	1	1.6	0.8	1	1
Speed_change_factor	1	1	1	1.5	0.75

Table 38 — HVXC object type reference output signal (continued)

File Name	HV02ref1	HV02ref2	HV03ref1	HV03ref2	HV04ref1	HV04ref2
test bitstream	HV02	HV02	HV03	HV03	HV04	HV04
Sampling rate[kHz]	8	8	8	8	8	8
Bitrate[kbit/s]	4	4	2	2	4	4
Variable rate	No	No	No	No	No	No
Delay mode	normal	low	normal	low	normal	low
Pitch_change_factor	1	1	1	1	1	1
Speed_change_factor	1	1	1	1	1	1

Table 39 — HVXC object type reference output signal (continued)

File Name	HV05ref1	HV05ref2	HV05ref3	HV05ref4	HV05ref5
test bitstream	HV05	HV05	HV05	HV05	HV05
Sampling rate[kHz]	8	8	8	8	8
Bitrate[kbit/s]	2	2	2	2	2
Variable rate	No	No	No	No	No
Delay mode	normal	normal	normal	normal	normal
Pitch_change_factor	1	1.6	0.8	1	1
Speed_change_factor	1	1	1	1.5	0.75
test_mode	7	7	7	15	15

Table 40 — HVXC object type reference output signal (continued)

File Name	HV05ref6	HV05ref7	HV05ref8	HV05ref9	HV05ref10
test bitstream	HV05	HV05	HV05	HV05	HV05
Sampling rate[kHz]	8	8	8	8	8
Bitrate[kbit/s]	2	2	2	2	2
Variable rate	No	No	No	No	No
Delay mode	low	low	low	low	low
Pitch_change_factor	1	1.6	0.8	1	1
Speed_change_factor	1	1	1	1.5	0.75
test_mode	7	7	7	15	15

Table 41 — HVXC object type reference output signal (continued)

File Name	HV06ref1	HV06ref2	HV07ref1	HV07ref2
test bitstream	HV06	HV06	HV07	HV07
Sampling rate[kHz]	8	8	8	8
Bitrate[kbit/s]	4	4	-	-
Variable rate	No	No	Yes	Yes
Delay mode	normal	low	normal	low
Pitch_change_factor	1	1	1	1
Speed_change_factor	1	1	1	1
test_mode	7	7	15	15

7.10 ER HVXC

The ER HVXC object is supported by the parametric speech coding (HVXC) tools with error resilient syntax, which provides fixed bitrate modes (2.0-4.0kbit/s) in a scalable and a non-scalable scheme, a variable bitrate modes (< 2.0kbit/s, <4.0kbit/s) and the functionalities of pitch and speed change. Only 8 kHz sampling rate and mono audio channel are supported.

7.10.1 Compressed Data

7.10.1.1 Characteristics

Encoders may apply restrictions to the following parameters of the compressed data.

7.10.1.1.1 AudioSpecificConfig

Compressed data provider must apply restrictions to the following parameters of the AudioSpecificConfig:

AudioObjectType

samplingFrequencyIndex

channelConfiguration

extensionFlag

epConfig

7.10.1.1.2 Bitstream payload

None.

7.10.1.1.3 BIFS/AudioSource node

A conform decoder may support any of the modification of the same parameters used in HVXC object type (see 7.9.3).

7.10.1.2 Test procedure

The AudioSpecificConfig must comply with the semantic conditions described below.

Table 42 — AudioSpecificConfig Characteristics

Syntactic element	Description
AudioObjectType	This field must be 25(ER HVXC object type).
SamplingFrequencyIndex	This field must be 0xb(8000Hz).
ChannelConfiguration	This field must be 1 (mono).
extensionFlag	This field must be 1 (error resilient syntax).
epConfig	No restrictions applay.

No restrictions apply to the Audio Access Unit.

7.10.2 Decoders

7.10.2.1 Characteristics

A conforming decoder may support any of the following modifications of some parameters in audio compressed data.

bitrate

variable rate (fixed rate/variable rate)

var_ScalableFlag (scalable/non-scalable mode in 4kbps variable rate)

A conforming decoder shall support one or both of the delay mode (normal delay mode/short delay mode), where the delay mode does not exist in audio compressed data.

7.10.2.2 Test procedure

Decoder conformance of the following modes is tested.

- error resilient syntax(epConfig=0/1)
- HVXC 4kbps variable rate mode

1. Procedures without a control interface

   1.1. All Voiced Compressed data …er_hv01_ep0(1) and er_hv02_ep0(1)

   1.2. All Unvoiced Compressed data (Direct Comparison by measuring segmental SNR) …er_hv03_ep0(1) and er_hv04_ep0(1)

2. Procedures with a control interface

   2.1. Direct Comparison by measuring segmental SNR(with random number generators disabled)…er_hv05_ep0(1), er_hv06_ep0(1), er_hv07_ep0(1), er_hv09_ep0(1) and er_hv10_ep0(1)

   2.2. Harmonic Phase Initialization (verification of phase randomness) …er_hv08_ep0(1)

Each test compressed data from er_hv01_ep0(1) to er_hv08_ep0(1) is identical to test compressed data of HVXC object from HV01 to HV08 except that they are error resilient syntax(re-ordered syntax). Therefore procedures to test decoder conformance of ER HVXC object are same as those of HVXC object and the corresponding reference waveforms are used. Details of procedures are referred in 7.9.6.

Regarding test compressed data er_hv09_ep0(1), newly added for HVXC 4kbps variable rate mode, er_hv09_nd and er_hv09_ld are used as reference waveforms. er_hv10_ep0(1) is a (2+2)[kbps] scalable compressed data. er_hv10_lay0_nd and er_hv10_lay0_ld are reference waveforms obtained by decoding only

base layer and er_hv10_lay1_nd and er_hv10_lay1_ld are reference waveforms obtained by decoding both base and enhancement layer.

7.10.2.3 Test sequences

Table 43 — ER HVXC Object Type Test Compressed data

File base name	er_hv01_ep0(1)	er_hv02_ep0(1)	er_hv03_ep0(1)	er_hv04_ep0(1)
Reference waveforms	HV01ref1~ HV01ref10	HV02ref1 HV02ref2	HV03ref1 HV03ref2	HV04ref1 HV04ref2
Content	All Voiced	All Voiced	All Unvoiced	All Unvoiced
Sampling Rate[kHz]	8	8	8	8
Bit Rate[kbit/s]	2	4	2	4
Variable Rate	No	No	No	No
Bitrate scalability	No	No	No	No
Number of enhancement layer	0	0	0	0
epConfig	0(1)	0(1)	0(1)	0(1)

Table 44 — ER HVXC Object Type Test Compressed data

File base name	er_hv05_ep 0(1)	er_hv06_ep 0(1)	er_hv07_ep0 (1)	er_hv08_ep0 (1)	er_hv09_ep0 (1)	er_hv10_ep0(1)
Reference waveforms	HV05ref1~ HV05ref10	HV06ref1 HV06ref2	HV07ref1 HV07ref2		er_hv09_nd er_hv09_ld	er_hv10_lay0 _nd er_hv10_lay0 _ld er_hv10_lay1 _nd er_hv10_lay1 _ld
Content						
Sampling Rate[kHz]	8	8	8	8	8	8
Bit Rate[kbit/s]	2	4	2	2	4	2+2
Variable Rate	No	No	Yes	No	Yes	No
Bitrate scalability	No	No	No	No	No	Yes
Number of enhancement layer	0	0	0	0	0	1
epConfig	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)

Table 45 — ER HVXC Object Type Reference waveforms

File base name	er_hv09_nd	er_hv09_ld	er_hv10_lay0_nd er_hv10_lay1_nd	er_hv10_lay0_ld er_hv10_lay1_ld
Content				
Samping Rate[kHz]	8	8	8	8
Bit Rate[kbit/s]	4	4	2+2	2+2
Variable Rate	Yes	Yes	No	No
Delay mode	Normal	Low	Normal	Low
Pitch_change_factor	1	1	1	1
Speed_change_factor	1	1	1	1
test_mode	15	15	15	15

7.11 ER HILN and ER Parametric

HILN tools are used in two object types: the Error Resilient HILN object type (ER-HILN) and the Error Resilient Parametric object type (ER-Parametric). If not stated otherwise the conformance criteria apply to both object types.

7.11.1 Compressed Data

7.11.1.1 Characteristics

7.11.1.1.1 AudioSpecificConfig

For a conforming compressed data, the following restrictions apply to syntactic elements of AudioSpecificConfig:

Elements of AudioSpecificConfig for the ER-HILN object type:

Table 46

audioObjectType	=26
samplingFrequencyIndex	in Natural Audio profile Level 1 or 3: 0x3 ... 0xc or =0xf in Natural Audio profile Level 2 or 4: ≤0xc or =0xf
samplingFrequency	in Natural Audio profile Level 1 or 3: ≤48000
channelConfiguration	=1

Elements of AudioSpecificConfig for the ER-Parametric object type:

Table 47

audioObjectType	=27
samplingFrequencyIndex	=0xb
channelConfiguration	=1

Elements of PARAconfig for the ER-HILN object type:

Table 48

PARAmode	=1
extensionFlag	=0

Elements of PARAconfig for the ER-Parametric object type:

Table 49

extensionFlag	=0

Elements of HILNconfig for object type ER-HILN:

Table 50

HILNsampleRateCode	in Natural Audio profile Level 1 or 3: 3 ... 12 in Natural Audio profile Level 2 or 4: ≤12
HILNframeLength	≥16
HILNcontMode	≤2

Elements of HILNconfig for object type ER-Parametric:

Table 51

HILNsampleRateCode	=11
HILNframeLength	=320
HILNcontMode	≤2

Elements of ErHVXCconfig for object type ER-Parametric: see 7.10

The maximum number of HILN extension layers is 7.

7.11.1.1.2 Bitstream payload

For a conforming compressed data, the following restrictions apply to syntactic elements of the audio access units:

The PCU value for computational complexity, which depends on the sampling frequency and the total number of sinusoids to be synthesized, must never exceed the value specified for a given profile and level as described in 1.5.2.3 of ISO/IEC 14496-3:2009. The calculation of PCU values for HILN is described in 1.5.2.2 of ISO/IEC 14496-3:2009. If HILN is used in HVXC/HILN mixed mode the HVXC PCU has to be added to the HILN PCU. If HILN is used in HVXC/HILN switched mode the HVXC PCU has to be added to the HILN PCU in all frames where a transition to or from HVXC occurs.

The following restrictions apply to elements of HILN audio access units:

Table 52

numLine	0 ... HILNmaxNumLine
numHarmPhase	0 ... numHarmLineTable[numHarmLineIndex]
numLinePhase	0 ... maxNumLinePhase (see calculation below)

Calculation of maxNumLinePhase:

```
maxNumLinePhase = 0;
for (i=0; i<numLine; i++)
    if (!linePred[i])
        maxNumLinePhase++;
```

For frequency and amplitude parameters, which are transmitted relative to the previously decoded values, the decoded values must never exceed the following limits:

Table 53

ILFreqIndex	0 ... maxFIndex[HILNsampleRateCode]-1
ILAmplIndex	0 ... 127
HarmFreqIndex	0 ... 2047
HarmAmplIndex	0 ... 127
NoiseAmplIndex	0 ... 127

7.11.1.2 Test procedure

A conforming compressed data must comply to the restrictions specified in 7.1.1.1. To test the conformance of a compressed data, it has to be parsed, the parameters have to be decoded, and the criteria have to be checked in every frame.

A modified version of the reference software allows to check for the described restrictions.

7.11.2 Decoders

7.11.2.1 Characteristics

The specifications given in the level definitions must be met. The output signal of the decoder under test must meet the criteria for accuracy of deterministic signal components and statistical properties of stochastic signal components as described below. Two alternative accuracy classes for HILN decoders are defined:

Full Accuracy HILN decoder

Fixed Point Accuracy HILN decoder

The criteria for deterministic signal components depend on the accuracy class selected. The criteria for stochastic signal components are the same for both accuracy classes.

7.11.2.2 Test procedure

Test compressed data and reference decoder output signals are provided to apply the different conformance criteria using the procedures described in the following Subclauses. Software implementing the different test procedures is available.

The ER-HILN and the ER-Parametric object types are both used only in the “Natural Audio” Profile. Since Level 2 includes Level 1, a Level 2 conforming decoder must also meet the criteria for Level 1. A Level 3 conforming decoder must also meet the criteria for Level 1, and a Level 4 conforming decoder must also meet the criteria for Levels 1, 2, and 3.

Since the conformance of the HILN decoder tools can be checked with compressed data for the ER-HILN object type and the conformance of the HVXC decoder tools can be checked with the compressed data for the ER-HVXC object type, compressed data for the ER-Parametric object type only tests operation of the integrated parametric decoder in its 4 different modes.

To be called a conforming ER-HILN or ER-Parametric decoder, the required conformance criteria must be met for all test compressed data listed in 7.11.2.3 that are applicable at the selected Profile and Level. The conformance criteria for deterministic signal components depend on the selected accuracy class.

Note: Due to the stochastic nature of the decoder characteristics tested in 7.11.2.2.3 to 7.11.2.2.5, there is a very small probability (less than 0.01) that such a test may fail for a conforming decoder. In this case, the test may be repeated with another initial state of the random number generator used in the decoder under test.

7.11.2.2.1 Conformance criterion for deterministic components of Full Accuracy HILN decoders

A Full Accuracy ER HILN decoder at an accuracy level of "K bit" has to fulfill the RMS/LSB criterion as defined in 7.1.2.2.1.

7.11.2.2.2 Conformance criteria for deterministic components of Fixed-Point Accuracy HILN decoders

The conformance criteria for Fixed-Point Accuracy decoders are based on measuring the segmental SNR and the LPC cepstral distortion (CD) between the reference decoder output and the output of the decoder to be tested. The segment length to be used in the calculation of the SNR is equal to the audio frame length as given in the compressed data tables below. The SNR and the CD have to be calculated only for the segments of which the power of the reference waveform is in the range [-50...-15] dB. CD is defined as

$$CD = \frac{10}{\ln(10)} \cdot \sqrt{2D}$$

D is the accumulated distortion of the LPC cepstrum C_{ref} of the reference waveform and C_{test} of the output of the decoder under test. D is defined as

$$D = \sum_{i=1}^{N} \left(C_{ref}[i] - C_{test}[i] \right)^2$$

N is the LPC cepstrum order which equals 32. The LPC cepstrum C[i] is defined by means of the algorithm `lpc2cepstrum` based on the LPC coefficients of a 16th order linear prediction filter. The computation of the LPC filter coefficients lpc_coef [j] is defined by the algorithm `calculate_lpc` (as defined in ISO/IEC 14496-3).

For a Fixed-Point Accuracy HILN decoder, the average value of the segmental SNR shall exceed 30 dB. At the same time, the average value of the CD shall not exceed 1 dB.

7.11.2.2.3 Conformance criteria for noise generators and spectral noise envelopes of HILN decoders

Noise components must not show a periodicity of less than one second. The average spectral envelope of a stationary noise component must meet a cepstral distance criterion when compared to the reference spectral envelope.

To perform this test, the noise signal is re-whitened by a prediction filter which is inverse to the filter used in the noise synthesis of the decoder. The required filter parameters are given in the parameter file accompanying a test compressed data. The autocorrelation function (ACF) of the re-whitened noise is calculated over a sufficiently long noise signal (e.g. 8 seconds, i.e. 256 frames) and normalized by dividing all values by the zero-th value of the ACF. The absolute values of this normalized ACF must not exceed a limit of e.g. 0.1 in the range of e.g. 1...15999.

In addition, the average power of the analyzed segment must be in the range of e.g. +/- 0.5 dB relative to the given reference power.

The detailed procedure is described in 7.11.2.2.6. The output signal of the decoder under test consists of different sections. For each section, starting at sample startpos and containing nframes*framelen samples, the test is initiated by invoking the function

```
test_noise(long nframes, long framelen, double *h, long order, long acflen,
        double aref, double max_noiseacf, double max_noiseadiff)
```

with the function parameters set according to the values given in the corresponding section of the parameter file accompanying the test compressed data. An example of one section of a parameter file is given below:

```
startpos          0
nframes           256
framelen          512
order             4
h[0]              0.78
h[1]              -0.49
h[2]              0.13
h[3]              -0.17
acflen            16000
aref              7550
max_noiseacf      0.1
max_noiseadiff    0.5
```

The procedures described in this Subclause are also used to test decoder output signals containing both deterministic and stochastic signal components. To perform this test, first the reference waveform – which contains only the deterministic signal components – is subtracted from the output of the decoder under test, and then the residual signal is assessed using the criteria for noise generators and spectral noise envelope described here.

7.11.2.2.4 Conformance criteria for temporal noise envelopes of HILN decoders

The average of multiple instances of the same temporal envelope must closely enough resemble the reference temporal envelope.

To perform this test, the signal is cut into segments with a length of 2 frames. For every sample position in this set of segments, the squares of the sample values of all segments are accumulated and afterwards divided by the number of segments to calculate the average power for each sample position. The square roots of the resulting values must not differ from the reference temporal amplitude envelope by more than e.g. +/-20% of the nominal noise amplitude.

The detailed procedure is described in 7.11.2.2.6. The output signal of the decoder under test consists of different sections. For each section, starting at sample startpos and containing nframes*framelen samples, the test is initiated by invoking the function

```
    test_noise_temporal_envelope(long nframes, long framelen,
                  double aref, long envcode, double max_noise_envdiff)
```

with the function parameters set according to the values given in the corresponding section of the parameter file accompanying the test compressed data. An example of one section of a parameter file is given below:

```
startpos              0
nframes               256
framelen              512
aref                  7000
envcode               0x676
max_noise_envdiff     20
```

7.11.2.2.5 Conformance criteria for random start phases of HILN decoders

The random start phases of individual lines and of the sinusoids in harmonic components must closely enough approximate a uniform distribution in the interval –pi ...pi and must show sufficient statistical independence of each other.

To perform this test, a signal that contains in an alternating way frames with no components and frames with some (e.g. 16...64) new beginning individual or harmonic lines with random start phases, is cut into segments with a length of 2 frames. The first segment must start at the center of the first frame, i.e. the sample where the fade-in of the synthesized sinusoids in the second frame starts.

For each segment, the amplitude and phase for every synthesized line is calculated. This is done by means of correlation with a windowed complex harmonic reference waveform

ref[i](t) = exp(j*2*pi*f[i]*t)*window(t)

at the frequencies f[i] of the synthesized lines as given in the corresponding parameter file (*.ctp).

The range of the phases (-pi...pi) is partitioned into 16 intervals of equal width. For each interval the number of phase parameters p[i] within this interval is determined. After e.g. 300 segments of 2 frames (i.e. 600 frames) the number of phase occurrences for each area must be in the range of e.g. 250...350.

Additionally a normalized ACF is calculated for all phases p[i] corresponding to 7.11.2.2.3. The absolute values of this ACF must not exceed a limit of e.g. 0.1 in the range of e.g. 1...499.

The detailed procedure is described in 7.11.2.2.6. The output signal of the decoder under test consists of different sections. For each section, starting at sample startpos and containing nframes*framelen samples, the test is initiated by invoking the function

```
test_startphase(long nframes, long framelen, double *fref, double *aref,
        long numline, long acflen, double samrate,
        long histmin, long histmax,
        double max_il_amplerr, double min_il_snr, double max_phaseacf)
```

with the function parameters set according to the values given in the corresponding section of the parameter file accompanying the test compressed data. An example of one section of a parameter file is given below:

```
startpos        0
nframes         600
framelen        512
samrate         16000
numline         4
freq[0]         101.6
freq[1]         201.6
freq[2]         298.4
freq[3]         398.4
aref[0]         950.5
aref[1]         950.5
aref[2]         950.5
aref[3]         950.5
acflen          500
histmin         250
histmax         350
max_il_amplerr  4
min_il_snr      30
max_phaseacf    0.1
```

7.11.2.2.6 Implementation of test procedures for stochastic signal components

Below the detailed description of the test procedures for stochastic signal components is given as pseudo-C code. The function get_next_sample() returns the value of the next sample of the output signal of the decoder under test. The function fail() is called whenever a conformance test failed, the parameter indicates the reason of the test failure.

```
#define pi  3.1415926535897932
#define zpi 6.2831853071795865  /* two pi   */

#define sqa(x)  ((x.re*x.re)+(x.im*x.im))
#define arc(x)  atan2(x.im, x.re)
#define abs(x)  sqrt(sqa(x))

typedef struct {double re, im;} complex;


void in_place_acf(double *x,long lxp)
{
    double  *tmp;
    long    i,j,n,n2;
    double  h;

    n=1<<lxp;
    n2=n/2;
    tmp=(double *) malloc(n2*sizeof(double));
    for (i=0; i<n2; i++){
        h=0.0;
        for (j=0; j<n2; j++) h+=x[j]*x[i+j];
        tmp[i]=h;
    }
    for (i=0; i<n2; i++) x[i]=tmp[i]/tmp[0];
    free(tmp);
}

/* this function finds the maximum absolute value in a given array
   and returns it's index   */

long findabsmax(double *x,long n)
{
    long    i,j;
    double  h,max;

    max=fabs(x[0]);
    j=0;
    for (i=1; i<n; i++){
        h=fabs(x[i]);
        if (  h>max ) { j=i; max= h; }
    }
    return j;
}

/* Pseudo-C Algorithm for testing the noise spectral envelope */

void test_noise(long nframes, long framelen, double *h, long order, long acflen,
        double aref, double max_noiseacf, double max_noiseadiff)
{
    double  *x;
    long    fx,fn,i,j;
    double  s,pwr,amp;

    fn=nframes*framelen;
    for (fx=1; (1<<fx)<2*fn; fx++);
    fn=1<<fx;               /* calculate buffer length   */

    x=(double *) malloc(fn*sizeof(double));
    pwr=0.0;
    for (i=0; i<order; i++){
```

```
        x[i]=0.0;
    }
    for (i=0; i<nframes*framelen; i++){
        s = get_next_sample();
        pwr+=s*s;
        x[i+order]=s;
        for (j=0; j<order; j++) s-=h[j]*x[order-1+i-j];     /* re-whitening  */
        x[i]=s;
    }
    for (; i<fn; i++) x[i]=0.0;             /* zero padding  */

    amp=sqrt(pwr/((double) (framelen*nframes)));

    in_place_acf(x,fx);     /* calculate acf in place, len=1<<fx   */

    i=findabsmax(x+1,acflen-1)+1;

    adiff=20.0*log10(amp/aref);
    fprintf(stderr,"noise ampl. diff: %10.5f dB\n",adiff);
    fprintf(stderr,"max. noise acf  : %10.5f at %i\n\n",x[i],i);

    if ( fabs(x[i]) > max_noiseacf   )
        fail("noise acf too big");
    if ( fabs(adiff) > max_noiseadiff )
        fail("invalid noise amplitude");
    free(x);
}

/* calculates the maximum difference between measured and reference envelope */

double env_diff(double *x, long env, long framelen)
{
    double  t0,ra,rd,t,y,d,dmax;
    long    i;

    t0=((((double)  (env>>8)    )+0.5)/16.0);
    t0+=1.125;
    ra=((((double) ((env>>4)&15))-0.5)/15.0);
    if ( ra<0.0 ) ra=0.0; else ra=5.0*tan(0.5*pi*ra);
    rd=((((double) ( env    &15))-0.5)/15.0);
    if ( rd<0.0 ) rd=0.0; else rd=5.0*tan(0.5*pi*rd);
    dmax=0.0;
    for (i=0; i<2*framelen; i++){
        t=(((double) i)+0.5)*(1.0/((double) framelen));
        if ( t<0.25 ) y=sin(zpi*t); else          /* fade in    */
        if ( t<1.00 ) y=1.0; else       /* hold       */
        if ( t<1.25 ) y=sin(zpi*(1.25-t)); else       /* fade out   */
        {                       /* given envelope */
            if ( t<t0 ) y=1.0-ra*(t0-t); else y=1.0-rd*(t-t0);
            if ( y<0.0 ) y=0.0;
        }
        d=fabs(x[i]-y);
        if ( d>dmax ) dmax=d;
    }
    return dmax;
}

/* Pseudo-C Algorithm for testing the noise temporal envelope */

void test_noise_temporal_envelope(long nframes, long framelen,
                    double aref, long envcode, double max_noise_envdiff)
```

```
{
    double      *pwrhist;
    long    i,j,k;
    double  s,mediff;

    pwrhist=(double *) malloc(2*framlen*sizeof(double));
    fprintf(stderr,"Testing %li frames of noise with envelope\n",nframes);
    for (i=0; i<2*framelen; i++) pwrhist[i]=0.0;
    k=framelen/8;
    for (i=0; i<framelen*nframes; i++){
        s = get_next_sample();
        pwrhist[k++]+=s*s;
        if ( k==2*framelen ) k=0;
    }
    for (i=0; i<2*framelen; i++){
        s=2.0*pwrhist[i]/((double) nframes);
        pwrhist[i]=sqrt(s)/aref;     /* normalized amplitude curve */
    }
    mediff = 100.0 * env_diff(pwrhist,envcode,framelen);
    /* calculate max. difference to reference envelope   */
    fprintf(stderr,"max. env. diff  : %10.5f %%\n\n",mediff);
    if ( mediff > max_noise_envdiff ) fail("noise envelope difference too big");
    free (pwrhist);
}

void cs_corr(complex *c,double *x,complex *s,long n)
{
    long    i;
    double  t,a,b,re,im;
    double  cc,ss,cs;

    re=im=cc=ss=cs=0.0;
    for (i=0; i<n; i++){
        a=s[i].re;
        b=s[i].im;
        re+=a*x[i];
        im+=b*x[i];
        cc+=a*a;
        ss+=b*b;
        cs+=a*b;
    }
    t=cc*ss-cs*cs;
    c->re=(re*ss-im*cs)/t;
    c->im=(im*cc-re*cs)/t;
}

/* Pseudo-C Algorithm for testing the random start phases */

void test_startphase(long nframes, long framelen, double *fref, double *aref,
                long numline, long acflen, double samrate,
                long histmin, long histmax,
                double max_il_amplerr, double min_il_snr, double max_phaseacf)
{
    complex     *camp;
    double      *ibuf;
    complex     *sbuf;
    double      *pacf;
    double      *pbuf;
    int     phist[16];
    complex     cr;
    double  f,a,p,h,ps,pn,mda,minsnr;
```

```
long    i,j,k,ph0,ph1,frnum,idx;

camp=(complex *) malloc(numline*sizeof(complex));
ibuf=(double *) malloc(2*framlen*sizeof(double));
sbuf=(complex *) malloc(numline*2*framelen*sizeof(complex));
pacf=(double  *) malloc(acflen*sizeof(double));
pbuf=(double  *) malloc(acflen*sizeof(double));

for (j=0; j<numline; j++){
    f=fref[j]*zpi/samrate;
    for (i=0; i<2*framelen; i++){
        p=((double) (i-framelen))+0.5;
        h=0.5+0.5*cos(p*(pi/((double) framelen)));
        sbuf[2*framelen*j+i].re=h*cos(f*p);
        sbuf[2*framelen*j+i].im=h*sin(f*p);
    }
}

mda=0.0;
minsnr=1000.0;

for (i=0; i<16; i++) phist[i]=0;
for (i=0; i<acflen; i++) pacf[i]=pbuf[i]=0.0;
for (i=0; i<framelen/2; i++) get_next_sample();
for (frnum=idx=0; frnum<nframes-2; frnum+=2){
    for (i=0; i<2*framelen; i++) ibuf[i] = get_next_sample();
    for (i=0,ps=0.0; i<2*framelen; i++) ps+=ibuf[i]*ibuf[i];
    for (j=0; j<numline; j++){
        cs_corr(camp+j,ibuf,sbuf+2*framelen*j,2*framelen);
        cr=camp[j];
        for (i=0; i<2*framelen; i++){
            ibuf[i] -= cr.re*sbuf[2*framelen*j+i].re +
                cr.im*sbuf[2*framelen*j+i].im;
        }
    }
    for (j=0; j<numline; j++){
        cr=camp[j];
        cs_corr(camp+j,ibuf,sbuf+2*framelen*j,2*framelen);
        cr.re+=camp[j].re;
        cr.im+=camp[j].im;
        a=abs(cr);  h=fabs(a-aref[j]); if ( h>mda ) mda=h;
        p=arg(cr);  phist[((long) ((p+pi)*(16.0/zpi)))&15]++;
        pbuf[idx]=p;
        for (i=0,k=idx; i<acflen; i++){
            pacf[i]+=p*pbuf[k];
            if ( k==0 ) k=acflen;
            k--;
        }
        cr=camp[j];
        for (i=0; i<2*framelen; i++){
            ibuf[i] -= cr.re*sbuf[2*framelen*j+i].re +
                cr.im*sbuf[2*framelen*j+i].im;
        }
        idx++;
        if ( idx==acflen ) idx=0;
    }
    for (i=0,pn=0.0; i<2*framelen; i++) pn+=ibuf[i]*ibuf[i];
    if ( (pn>0.0) && (ps>0.0) ){
        h=10.0*log10(ps/pn);
        if ( h<minsnr ) minsnr=h;
    }
```

```
    }
    for (i=frnum*framelen+framelen/2; i<nframes*framelen; i++) get_next_sample();
    free(sbuf);
    free(camp);
    i=findabsmax(pacf+1,acflen-1)+1;
    a=pacf[i]/pacf[0];
    fprintf(stderr,"max. ampl. diff.: %10.5f lsb\n",mda);
    fprintf(stderr,"min. SNR        : %10.5f dB\n",minsnr);
    fprintf(stderr,"max. startp. acf: %10.5f at %i\n\n",a,i);
    ph0=ph1=0;
    j=phist[0];
    for (i=1; i<16; i++){
        k=phist[i];
        if ( k<phist[ph0] ) ph0=i;
        if ( k>phist[ph1] ) ph1=i;
        j+=k;
    }
    fprintf(stderr,"min. norm. phist: %f at %i\n",
        16.0*((double) phist[ph0])/((double) j),ph0);
    fprintf(stderr,"max. norm. phist: %f at %i\n\n",
        16.0*((double) phist[ph1])/((double) j),ph1);
    if ( phist[ph0] < histmin || phist[ph1] > histmax )
        fail("incorrect start phase distribution");
    if ( mda     > max_il_amplerr )
        fail("incorrect individual line amplitude");
    if ( minsnr  < min_il_snr     )
        fail("too much noise in individual line synthesis");
    if ( fabs(a) > max_phaseacf   )
        fail("start phase periodicity detected");
    free(pacf);
    free(pbuf);
    free(ibuf);
}
```

7.11.2.3 Test sequences

Table 54 through

Table 57 below describe the conformance test compressed data for ER-HILN and ER-Parametric object types for the different levels of the Natural Audio Profile.

Conformance criteria abbreviations:

determin — see 7.11.2.2.1 or 7.11.2.2.2 according to selected accuracy class

noise gen — see 7.11.2.2.3 [1])

noise env — see 7.11.2.2.4 [1])

start phase — see 7.11.2.2.5 [1])

det+noise — first the reference waveform (containing only the deterministic components) is subtracted and then test procedure described in 7.11.2.2.3 is applied to the residual signal obtained by the subtraction

subjective — no objective assessment necessary [1])

[1]) reference waveforms are supplied for subjective assessment

The additional functionality of HILN speed change and pitch change is tested using the compressed data er_hi03 and er_hi21 together with the reference waveforms (*.wav) / reference parameter files (*.ctp) given in

Table 57.

Table 54 — ER-HILN Object Type Test sequences for Natural Audio Profile Level 1

File base name	er_hi00_ep0	er_hi01_ep0	er_hi02_ep0	er_hi03_ep0
Description	harm+indi	harm+indi	harm+indi	harm+indi
Conformance Criterion	determin	determin	determin	determin
Reference Waveform	er_hi00	er_hi01	er_hi02	er_hi03
Sampling Rate [kHz]	16	16	16	16
Frame Length [samples]	512	512	512	512
epConfig	0	0	0	0
HILNquantMode	0	0	0	1
HILNcontMode	0	1	2	2
Enhancement Layer				
Extension Layer(s)				
Max. Bit Rate [kbit/s]	16	16	16	16

File base name	er_hi04_ep0	er_hi05_ep0	er_hi06_ep0	er_hi07_ep0
Description	harm+indi	harm+indi	harm+indi	harm+indi
Conformance Criterion	determin	determin	determin	determin
Reference Waveform	er_hi04	er_hi05	er_hi06	er_hi07
Sampling Rate [kHz]	8	8	8	8
Frame Length [samples]	256	256	256	256
epConfig	0	0	0	0
HILNquantMode	0	0	0	1
HILNcontMode	0	1	2	2
Enhancement Layer				
Extension Layer(s)				
Max. Bit Rate [kbit/s]	6	6	6	6

File base name	er_hi10_ep0	er_hi11_ep0	er_hi12_ep0
Description	harm+indi	harm+indi	PCU = 20
Conformance Criterion	determin	determin	determin
Reference Waveform	er_hi10	er_hi11	er_hi12
Sampling Rate [kHz]	16	48	16
Frame Length [samples]	80	3072	1280
epConfig	0	0	0
HILNquantMode	1	1	1
HILNcontMode	2	2	2
Enhancement Layer			
Extension Layer(s)			
Max. Bit Rate [kbit/s]	32	32	32

File base name	er_hi13_ep0	er_hi14_ep0	er_hi15_ep0	er_hi16_ep0
Description	harm+indi	harm+indi	harm+indi	harm+indi
Conformance Criterion	lay0: subjective lay1: determin	lay0: subjective lay1: determin	determin	determin
Reference Waveform	er_hi13_lay0 er_hi13_lay1	er_hi14_lay0 er_hi14_lay1	er_hi15_lay0 er_hi15_lay1	er_hi16_lay0 er_hi16_lay1 er_hi16_lay2 er_hi16_lay3 er_hi16_lay4 er_hi16_lay5 er_hi16_lay6 er_hi16_lay7
Sampling Rate [kHz]	8	16	32	16
Frame Length [samples]	256	512	1024	512
epConfig	0	0	0	0
HILNquantMode	1	1	1	1
HILNcontMode	2	2	2	2
Enhancement Layer	1	1		
Extension Layer(s)			1	7
Max. Bit Rate [kbit/s]	6+6	16+16	6+10	6+2+2+2+2+2+2+2

File base name	er_hi20_ep0	er_hi21_ep0	er_hi22_ep0
Description	harm+indi	noise	noise
Conformance Criterion	start phase	noise gen	noise env
Reference Waveform Reference Parameter	er_hi20 er_hi20	er_hi21 er_hi21	er_hi22 er_hi22
Sampling Rate [kHz]	16	16	16
Frame Length [samples]	512	512	512
epConfig	0	0	0
HILNquantMode	1	1	1
HILNcontMode	2	2	2
Enhancement Layer			
Extension Layer(s)			
Max. Bit Rate [kbit/s]	6	6	6

File base name	er_hi23_ep0	er_hi24_ep0	er_hi25_ep0
Description	harm+indi	noise	noise
Conformance Criterion	start phase	noise gen	noise env
Reference Waveform Reference Parameter	er_hi23 er_hi23	er_hi24 er_hi24	er_hi25 er_hi25
Sampling Rate [kHz]	8	8	8
Frame Length [samples]	256	256	256
epConfig	0	0	0
HILNquantMode	1	1	1
HILNcontMode	2	2	2
Enhancement Layer			
Extension Layer(s)			
Max. Bit Rate [kbit/s]	6	6	6

File base name	er_hi26_ep0 er_hi26_ep1	er_hi27_ep0 er_hi27_ep1	er_hi28_ep0 er_hi28_ep1	er_hi29_ep0 er_hi29_ep1
epConfig	0,1	0,1	0,1	0,1
Description	harm+indi+noise	harm+indi+noise	music	music
Conformance Criterion	det+noise	det+noise	subjective	subjective
Reference Waveform Reference Parameter	er_hi26 er_hi26	er_hi27 er_hi27	er_hi28	er_hi29
Sampling Rate [kHz]	16	8	16	8
Frame Length [samples]	512	256	512	256
HILNquantMode	0	0	0	0
HILNcontMode	0	0	0	0
Enhancement Layer				
Extension Layer(s)				
Max. Bit Rate [kbit/s]	16	6	16	6

Table 55 — ER-HILN Object Type Test Compressed data for Natural Audio Profile Level 2

File base name	er_hi30_ep0	er_hi31_ep0
Description	harm+indi	PCU = 100
Conformance Criterion	determin	determin
Reference Waveform	er_hi30	er_hi31
Sampling Rate [kHz]	96	48
Frame Length [samples]	2048	3840
epConfig	0	0
HILNquantMode	1	1
HILNcontMode	2	2
Enhancement Layer		
Extension Layer(s)		
Max. Bit Rate [kbit/s]	32	32

Table 56 — ER-Parametric Object Type Test Compressed data for Natural Audio Profile Level 1

File base name	er_pa00_ep0	er_pa01_ep0	er_pa02_ep0	er_pa03_ep0
Description	HVXC only	HILN only	switched mode	mixed mode
Conformance Criterion	determin (fixed point accuracy)	determin (fixed point accuracy)	determin (fixed point accuracy)	determin (fixed point accuracy)
Reference Waveform	er_pa00	er_pa01	er_pa02	er_pa03
Sampling Rate [kHz]	8	8	8	8
Frame Length [samples]	160	256	320	320
epConfig	0	0	0	0
PARAmode	0	1	2	3
HILNquantMode		0	0	0
HILNcontMode		0	0	0
Enhancement Layer				
Extension Layer(s)				
Max. Bit Rate [kbit/s]	4	6	4	8

Table 57 — ER-HILN Reference Waveforms / Parameters for test of additional functionality

File base name	er_hi03_ep0	er_hi03_ep0	er_hi03_ep0	er_hi03_ep0
Description	harm+indi	harm+indi	harm+indi	harm+indi
Conformance Criterion	determin	determin	determin	determin
Reference Waveform	er_hi03_s08	er_hi03_s16	er_hi03_p07	er_hi03_p15
Sampling Rate [kHz]	16	16	16	16
Frame Length [samples]	512	512	512	512
epConfig	0	0	0	0
HILNquantMode	1	1	1	1
HILNcontMode	2	2	2	2
speedFactor	0.8	1.6	1	1
pitchFactor	1	1	0.7	1.5
Max. Bit Rate [kbit/s]	16	16	16	16

File base name	er_hi21_ep0	er_hi21_ep0	er_hi21_ep0	er_hi21_ep0
Description	noise	noise	noise	noise
Conformance Criterion	noise gen	noise gen	subjective	subjective
Reference Waveform Reference Parameter	er_hi21_s08 er_hi21_s08	er_hi21_s16 er_hi21_s16	er_hi21_p07	er_hi21_p15
Sampling Rate [kHz]	16	16	16	16
Frame Length [samples]	512	512	512	512
epConfig	0	0	0	0
HILNquantMode	1	1	1	1
HILNcontMode	2	2	2	2
speedFactor	0.8	1.6	1	1
pitchFactor	1	1	0.7	1.5
Max. Bit Rate [kbit/s]	16	16	16	16

File base name	er_hi28_ep0	er_hi28_ep0	er_hi28_ep0	er_hi28_ep0
Description	music	music	music	music
Conformance Criterion	subjective	subjective	subjective	subjective
Reference Waveform	er_hi28_s08	er_hi28_s16	er_hi28_p07	er_hi28_p15
Sampling Rate [kHz]	16	16	16	16
Frame Length [samples]	512	512	512	512
epConfig	0	0	0	0
HILNquantMode	0	0	0	0
HILNcontMode	0	0	0	0
speedFactor	0.8	1.6	1	1
pitchFactor	1	1	0.7	1.5
Max. Bit Rate [kbit/s]	16	16	16	16

7.12 TTSI

Conformance testing of the TTSI shall be performed by comparing the decoded parameters to those supplied with the corresponding bitstreams.

Since the MPEG-4 Audio TTSI described in ISO/IEC 14496-3 restricts its standardization subjects only to the interfaces as depicted in Figure 8 and the exact synthesis method is not standardized, the quality of synthesized speech will not be tested. Several interfaces should be tested among the interfaces in Figure 8 are distinguished in the Table 58.

Table 58 — Interfaces of MPEG-4 Audio TTSI

Number	Interface Name	Test?
6.1	Interface between DEMUX and the syntactic decoder	Yes
6.2	Interface between the syntactic decoder and the speech synthesizer	Yes
6.3	Interface from the speech synthesizer to the compositor	No
6.4	Interface from the compositor to the speech synthesizer	No
6.5	Interface between the speech synthesizer and the phoneme-to-FAP converter that is part of the Face node in ISO/IEC 14496-1	Yes

7.12.1 Object Descriptor Characteristics

Encoders may apply restrictions to the following parameters of the object descriptor:

a) Language_Code indicates flag of language identification that should be supported by TTS engine.
b) Gender_Enable indicates the existence of gender information.
c) Age_Enable indicates the existence of age information.
d) Speech_Rate_Enable indicates the existence of speech rate information.
e) Video_Enable Enable is set to “1” when TTS decoder works with MP.
f) Lip_Shape_Enable indicates the existence of lip shape information.
g) Trick_Mode_Enable indicates that trick mode function is allowed.

7.12.2 Elementary Stream Characteristics

TTS Encoder may apply restrictions to the following parameters of the bitstream:

a) language code and input text length
b) phoneme symbols

c) prosody information

d) lip shape pattern

e) gender and age of the speaker

7.12.3 Procedure to Test Bitstream Conformance

Verify the test bitstream is conformant with the bitstream syntax described in ISO/IEC 14496-3 subpart 6.

7.12.4 Decoder Characteristics

Decoder conformance test will be applied to several interfaces identified in the Table 58. For interface 6.1 and 6.2, a number of test bitstreams will be supplied by the electronic attachment to this part of ISO/IEC 14496 and it should be verified whether the syntax and semantics are correctly interpreted. For interface 6.5 it should be tested whether a speech synthesizer correctly generates defined data structure even if phoneme symbols and additional prosodic information are not available from the input bitstreams. When the information is included in input bitstream, conformance should be also tested whether the parameter values from the speech synthesizer to the Phoneme/Bookmark-to-FAP converter are equivalent to those supplied with corresponding bit streams.

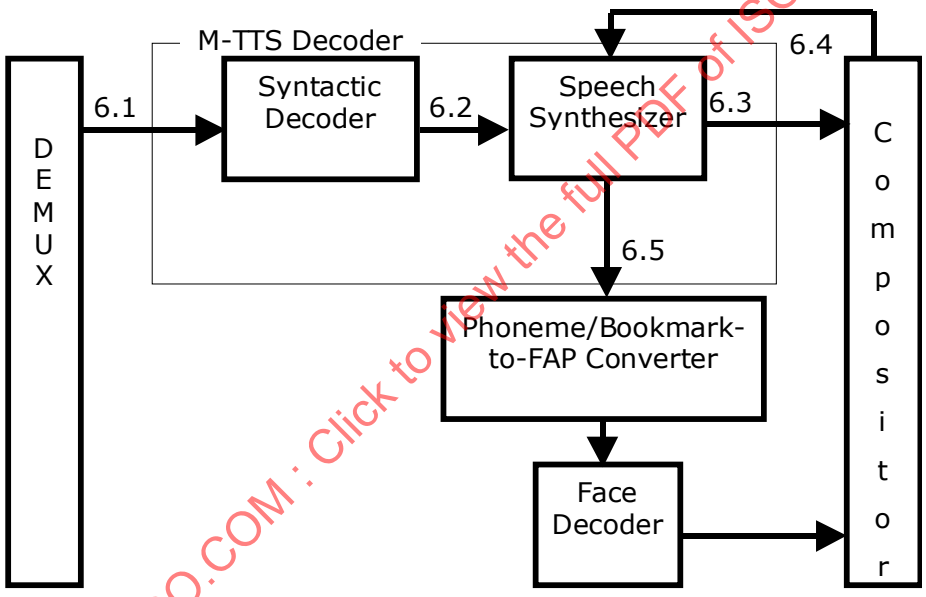

Figure 8 — MPEG-4 Audio TTS decoder architecture. (Text-to-Speech in ISO/IEC 14496-3)

7.12.5 Procedure to Test Decoder Conformance

To test audio decoders, the electronic attachment to this part of ISO/IEC 14496 supplies a number of test sequences. Supplied sequences cover Text with Language_Code only, and additional information such as Gender, Age, Speech_Rate, etc. The test set includes a set of additional information, as listed in Table 59. The extension number is appended to the bitstream name to indicate the additional information of the test sequence.

For a supplied test sequence, testing of Syntactic decoder (6.2) can be done by comparing the output of a decoder under test with a reference output also supplied by the electronic attachment to this part of ISO/IEC 14496. Interface between speech synthesizer and phoneme/bookmark-to-FAP converter (6.5) can be tested by comparing the output of a speech synthesizer under test with a reference output of TTS2 test bitstream in Table 59. The interface data includes phonemeSymbol, phonemeDuration, F0Average, stress,

and wordBegin information for each phoneme in a sentence. At the beginning or end of words, a bookmark may be associated with a phoneme. Especially, phonemeSymbol, phonemeDuration, F0Average, bookmark, and wordBegin information should be identical to the reference data; stress is TTS engine dependent. Software is provided for performing this verification procedure. Measurements are carried out by the exactness of output data of decoder. To be called an ISO/IEC 14496-3 audio decoder, the decoder shall provide an output such that accordance between the output of the decoder under test and the supplied reference output is achieved. This test only verifies the computational accuracy of an implementation. The streams TTS7 and TTS8 contain a TTS stream and a BIFS scenegraph with a Face node in order to evaluate the integration of face animation and TTSI verifying gender and animation of the face. Conformance is tested at interface 7.2 and the parameters of the FAP node.

7.12.6 Descriptions of the test bitstreams

ISO/IEC 14496-3 audio test bitstreams are suggested as follows:

Table 59 — TTS Object type Test Bitstream

File Name	TTS1	TTS2	TTS3	TTS4	TTS5	TTS6	TTS7	TTS8
Content								
Language_Code	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Text	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Gender	Yes						Yes	Yes
Age	Yes							
Speech_Rate	Yes							
Phoneme_Symbols		Yes			Yes	Yes		
Dur_each_Phonemes		Yes			Yes	Yes		
F0_Contour_each_Phoneme		Yes			Yes	Yes		
Energy_Contour_each_Phoneme		Yes						
Video_Enable			Yes		Yes			
Lip_Shape_Enable				Yes		Yes		

For the decoder conformance test for interface 6.2, all the test bitstreams should be verified. Also test bitstream TTS1, TTS4, TTS6, TTS7, and TTS8 should be verified for the conformance test of the interface between speech synthesizer and Phoneme/Bookmark-to-FAP converter.

7.13 General MIDI

The General MIDI object type supports General MIDI patch mappings. This object type provides backward-compatibility with existing MIDI content and rendering devices. Normative and implementation-independent sound quality cannot be produced in this object type.

7.13.1 Procedure to Test Bitstream Conformance

The bitstream syntax must first comply with the description given in 5.14 of subpart 5 of ISO/IEC 14496-3:2009.

7.13.1.1 DecoderSpecificInfo Characteristics

AudioObjectType: Shall be encoded with the value 15

SamplingFrequencyIndex: Shall be encoded with the following values:

The following restrictions apply to StructuredAudioSpecificConfig:

Only the **midi_file** chunk shall occur in the StructuredAudioSpecificConfig.

7.13.1.2 Audio Access Unit Characteristics

Any **SMF** and/or **midi** elements transmitted in Audio Access Units must comply with the Standard MIDIFIle Format 0 specification and MIDI protocol specification as normatively referenced in 5.5.2 of subpart 5 of ISO/IEC 14496-3:2009.

Only the **midi_event** event shall occur in the Audio Access Units.

7.13.2 Conformance

The General MIDI object type is included only to provide interoperability with existing content. Normative sound quality and decoder behaviour are not provided with the General MIDI object type. Refer "The Complete MIDI 1.0 Detailed Specification – Version 96.1" (c) 1996 MIDI Manufacturers Association for all aspects of bitstream and decoder conformance testing.

7.14 Wavetable Synthesis

This object type is used to describe music and sound-effects content in situations in which the full flexibility and functionality of SAOL, including 3-D audio, is not required. The wavetable synthesis object type incorporates only the SASBF format and MIDI tools. It allows the use of simple "sampling synthesis" in presentations where the quality and flexibility of the full synthesis toolset is not required.

Refer the following publications for all aspects of bitstream and decoder conformance testing:

- The Complete MIDI 1.0 Detailed Specification – Version 96.1 (c) 1996 MIDI Manufacturers Association
- The Downloadable Sounds Level 2 Specification – (c) 1999 MIDI Manufacturers Association
- The Downloadable Sounds Certification Test (exact title to be determined)

7.14.1 Procedure to Test Decoder Conformance

7.14.1.1 DecoderSpecificInfo Characteristics

AudioObjectType: Shall be encoded with the value 14

The following restrictions apply to StructuredAudioSpecificConfig:

Only the **midi_file** and **sbf** chunks shall occur in the StructuredAudioSpecificConfig.

7.14.1.2 Audio Access Unit Characteristics

The bitstream syntax must comply with the description given in 5.13 of subpart 5 of ISO/IEC 14496-3:2009. In addition:

Any **sasbf** elements transmitted in the Audio Access Units must comply with the DLS-2 syntax as normatively referenced in 5.2 of that subpart.

Only the **midi_event** event shall occur in the Audio Access Units.

7.14.2 Conformance

To facilitate interoperability between MPEG-4 SASBF implementations, as well as between implementations providing similar functionality using the MIDI Manufacturers Association "DLS-2" file format and synthesis model [DLS2], it is desirable to have a series of conformance tests which will allow equipment vendors to self-test for compliance with the SASBF specification. Such conformance tests should ideally test all aspects of the design to verify compliance with the specification. As a practical matter, it will not be possible to assure 100 % compliance with any finite series of tests, but a subset can be created which will provide a reasonable degree of certainty that a particular device is in compliance. Note that the terms SASBF and DLS-2 refer to the same

standard and are used interchangeably in this document and in the test materials. In particular, the abbreviation “dls” is more commonly used in tables, file names and so forth.

Please see Annex A for detailed information about the test bitstreams and other materials.

Each test has been devised to exercise a particular function of the implementation. By isolating functions, it is possible to identify specific failures easily. However there is a risk that interactions between functions within an implementation could result in complex failure modes. Without an intimate knowledge of the architecture of a specific implementation, it will be impossible to anticipate these complex failure modes. Therefore it is left to the designer to anticipate and device specific test scenarios for such complex failure modes.

The conformance tests comprise a SASBF instrument file, a corresponding MIDI file, and a sample output file in .wav format from the reference implementation. The sample output file is intended to be used as a comparison for correctness. The nature of MIDI and SASBF do not lend themselves to bit-for-bit comparisons, so more sophisticated methods of signal analysis will be necessary when significant variations are encountered between the implementation under test and the reference implementation.

7.15 Algorithmic Synthesis and AudioFX

The Algorithmic Synthesis object type provides SAOL-based synthesis capabilities for very low-bitrate terminals. It is also used to support the AudioBIFS AudioFX node when sound synthesis capability is not needed.

7.15.1 DecoderSpecificInfo Characteristics

Bitstream provider may apply restrictions to the following parameters of the DecoderSpecificInfo:

a) **orchestra** block

7.15.2 Audio Access Unit Characteristics

Bitstream provider may apply no restrictions to any parameters of the bitstream.

7.15.3 Procedure to Test Bitstream Conformance

7.15.3.1 DecoderSpecificInfo Characteristics

AudioObjectType: Shall be encoded with the value 16

SamplingFrequencyIndex: Shall be encoded with the following values:

Table 60

SamplingFrequencyIndex	Level 1	Level 2	Level 3	Level 4
Synthetic Audio Profile	>= 6	>= 3	0..0xc	
Main Profile	0..0xc			

SamplingFrequency: Shall be encoded with the following values:

Table 61

SamplingFrequency	Level 1	Level 2	Level 3	Level 4
Synthetic Audio Profile	<= 24000	<= 48000	<= 96000	
Main Profile	0..96000			

ChannelConfiguration: Shall be encoded with the following values:

Table 62

ChannelConfiguration	Level 1	Level 2	Level 3	Level 4
Synthetic Audio Profile	1	1..2	1..7	
Main Profile	1..7			

The following restrictions apply to StructuredAudioSpecificConfig:

orchestra: must comply with the SAOL syntax and rate rules.

7.15.3.2 Audio Access Unit Characteristics

score: any score lines transmitted in the access units must comply with the SASL syntax.

7.15.4 Decoder Characteristics

All signal variables in SAOL shall be represented by a 32-bit floating-point value as defined in ISO/IEC 14496-3:2009, subpart 5, 5.8.3. Implementations are free to use any internal representation for variable values, so long as the results calculated are identical to the results of the calculations using 32-bit floating-point values.

The order of execution of the Structured Audio primitives may be rearranged if it will have no effect on the output of the decoding process, i.e. if the output of the decoding process still satisfies the conformance criterion.

Some of the SAOL functionality is not testable and measurable on an operations-per-second basis, since some of the decoding algorithms for core opcodes and statements are not specified and left open to the implementers; among them, some like interpolation, spatialization, effects and filters could heavily affect allocated memory and computational complexity of a specific decoder. In conclusion, it is necessary to follow some macro-oriented criteria, which are able to make abstraction of the open issues, and calculate them in separate elements of a defined *complexity vector*. At the same time the complexity vector must not be too long, because this could hardly overspecify the decoder when the SAOL functionality is not completely exploited. The complexity vector is defined as follows:

[*total core opcode calls*, *floating-point operations*, *multiplications*, *tests*, *mathematical methods*, *noise generators*, *interpolations*, *multiply-and-add*, *filters*, *effects*, *allocated memory*].

Criteria to calculate the complexity vector are specified in Annex A. The Annex describes in details the method for measuring decoding complexity of normative MPEG-4 Structured Audio streams. This method provides metrics to define levels of the Structured Audio Object types 3 and 4, as far as possible in a platform independent and implementation independent manner. The Annex contains the principles to select the complexity vector and how to calculate it; then the software tool is presented, which is based on the Structured Audio decoder reference software. The complexity Measurement Tool for Level Definitions of Algorithmic Synthesis and AudioFX Object Type, and the corresponding profiler tool is provided by the electronic attachment to this part of ISO/IEC 14496.

Table 63, called «Algorithmic Synthesis Complexity Values for Levels»,specifies values for SA Object type 3 for algorithmic synthesis: they are used in Synthetic Audio Profile Level definitions to define “Low”, “Medium” and “High” Complexity values:

Table 63 — Algorithmic Synthesis Complexity Values for Levels

Parameter	Low Complexity	Medium Complexity	High Complexity
Total opcode calls	2M	8M	16M
Floating-point ops	12M	24M	60M
Multiplications	8M	16M	40M
Tests	2M	8M	16M
Math methods	4M	16M	16M
Noise generators	0.1 M	1M	1M
Interpolations	0.6 M	4M	12M
Multiply-and-add	2M	4M	12M
Filters	0.6M	2M	4M
Effects	0.2M	1M	2M
Allocated memory	64k	8M	16M

It is not the case that in order to conform to one of the complexity levels in the table that a decoder must provide the amount of computation shown in the table for every element of the complexity vector at the same time. Rather, a conforming decoder must be able to normatively decode any bitstream that is measured with the standard profiling tool as requiring no more than that amount of computation. When a conforming decoder is implemented with static optimization, it is usually possible to decode a bitstream that contains a certain number of operations per second as measured with the profiling tool by actually using many fewer operations per second than this, because the calculation of the complexity vector is made in a platform independent way on the basis of the normative SA text. Put another way, there are two ways to increase the amount of computation that a Structured Audio decoder can provide. On one hand, it can run on more powerful hardware. On the other, it can implement more powerful static optimization and thereby provide more effective computation on the same hardware. The measurements shown in the table should be taken as referencing a completely unoptimized SA implementation, and so high complexity decoding can actually be realized on a hardware platform without nearly so much native computational power. Each implementor should be able to "map" these platform independent formal vectors into his own implementation using Annex A, in order to calculate his actual complexity vectors.

Implementors are also advised that algorithmic synthetic bitstreams often require "bursty" processing, where small time portions of the bitstream require considerable amount of processing power. In situations such as this, where the requirements of a bitstream exceed in rare spikes of time (granularity of the profiling is 1 second) the complexity of a particular level, implementors are encouraged to implement a procedure for graceful degradation of decoding. Many such techniques exist, such as voice stealing, but they are non-normative and left up to the implementor. Priority bits are also provided to support such techniques (see 5.7.3.3.7 and 5.7.3.3.8 of ISO/IEC 14496-3:2009 subpart 5). Such techniques can also result in great benefit for the case of high degrees of user interaction, which could hardly affect the overall schedulability of the system.

Complexity values for the AudioFX node are specified in the following Table. For conformance test of the AudioFX node see also 9.1.10 and Table 64, where these values are used.

Table 64 — Complexity values for AudioFX node levels

Parameter	Very Low Complexity	Low Complexity	Medium Complexity	High Complexity
Total opcode calls	1M	1M	4M	8M
Floating-point operations	0	4M	12M	20M
Multiplications	0	2M	8M	16M
Tests	0	1M	4M	8M
Math methods	0	2M	6M	12M
Noise generators	0	0.05 M	0.2M	0.5M
Interpolations	0	0.3M	1.2M	2M
Multiply-and-add	2M	2M	4M	8M
Filters	0.2M	0.2M	1M	4M
Effects	96k	96k	0.4M	2M
Allocated memory	96k	96k	1M	16M

7.15.5 Procedure to Test Decoder Conformance

As with the natural audio coders, many functions of the Structured Audio decoder can be checked for conformance by RMS measurement of the residual after comparison to the reference signal. Other functions cannot use this criterion because either the decoding process uses functions of the decoder which are not strictly normative, or the decoding process depends on non-deterministic random number or noise generators as described in 5.9.8 and 5.10.4 of ISO/IEC 14496-3:2009 subpart 5.

Testing the deterministic, strictly normative functions shall be performed by comparing the output of a decoder under test with a reference output supplied by the electronic attachment to this part of ISO/IEC 14496 using the procedure described in 7.1.2.2.1. Software is provided for performing this verification procedure. Measurements are carried out relative to full scale where the output signals of the decoders are normalized to be in the range between –1 and +1. This test verifies the computational accuracy of an implementation. Conformant decoders must use the RMS Measurement criterion for sequences SY001 through SY004 and SY016 through SY019.

Bitstreams SY005 through SY009 use syntactic elements that are not strictly normative. Conformant decoders shall parse these bitstreams, but a test using RMS measurement is not possible in these cases. This last group of bitstreams is more oriented towards the test of overall complexity capabilities of the decoder. An implementation that claims conformance to any of the complexity levels within a profile must have the minimum capacity as shown in Table 63. See also 7.15.4, and 5.7.3.3.7 and 5.7.3.3.8 of ISO/IEC 14496-3:2009 subpart 5 for more details.

Decoder conformance concerning computation capabilities shall be tested against the definition of high, medium or low computational complexity provided in Table 63. The decoder supporting one of the three computational levels shall be able to decode bitstreams for which the associated complexity vector is, for each second of the performance, below the reference vector of the corresponding Level. Rare exceptions are admitted as explained in 7.15.4. The decoding time of each second of the performance shall be executed in a time less or equal to a wall clock second. Bitstreams SY005 through SY009 are provided by the electronic attachment to this part of ISO/IEC 14496 with their corresponding complexity vectors in function of time, in order to help the correct evaluation of the computational complexity supported by the specific decoder.

Testing of the non-normative interpolation (**interp** equal to 1 in the global block of the SAOL orchestra) shall be performed using test sequence SY010 and SY011. A reference output is provided by the electronic attachment to this part of ISO/IEC 14496. To be called an ISO/IEC 14496-3 audio decoder, the decoder shall provide an output for which the SNR between SY011 and the reference output is strictly less than the SNR between SY010 and the reference output. To calculate the SNR, the difference shall be calculated between the specified sequence output and the reference, and this difference shall be used as noise of the reference output.

Testing of the non-normative noise generators shall be performed using test sequence SY012. The output of the decoder shall be divided into 5 groups of 40000 samples, in order to isolate the 5 different types of noise generators, as described in 7.15.6. The sequence shall be repeated three times and the output analyzed separately.

To be called an ISO/IEC 14496-3 audio decoder, the decoder shall provide an output satisfying the following conditions:

a) samples generated with linear distribution shall have a mean value m such that -2*10exp-3 < m < 2*exp-03 and a variance v such that 0.3300 < v < 0.3366. These two constraints shall be met at least in two of the three repetitions.

b) samples generated with gaussian (normal) distribution shall have a mean value m such that -5*10exp-3 < m < 5*exp-03 and a variance v such that 0.5 < v < 0.5170. These two constraints shall be met at least in two of the three repetitions.

c) samples generated with linearly-ramped distribution shall be converted to a linear distribution using the formula: y = +-sqrt(x), where x is the generated vector and y is the resulting vector, obtained taking alternatively a positive and a negative value. The resulting vector y shall be evaluated as in a)

d) samples generated with exponential distribution shall have a mean value m such that 0.4300 < m < 0.4330. This constraint shall be met at least in two of the three repetitions.

e) binary samples generated with poissonian generators shall have a mean m value such that 0.4900 < m < 0.5100. This constraint shall be met at least in two of the three repetitions.

Testing of the non-normative **lopass**, **hipass**, **bandpass**, **bandstop** core opcodes shall be performed using test sequence SY013. The output of the sequence shall be divided in 4 sub-blocks of 16000 samples, corresponding to the test of the 4 filters above. The DFT of the four blocks shall be calculated, and the absolute value of the resulting spectrum shall be evaluated against the mask of Figure 9.

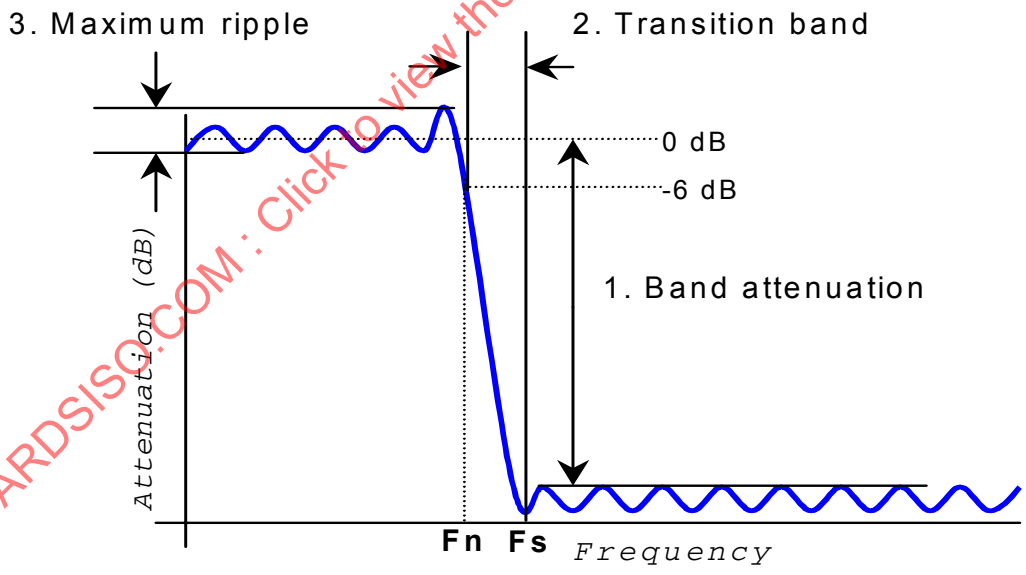

Figure 9

The maximum ripple is the absolute difference between the greatest and least response in the region limited by the -6 dB absolute value (pass band region). The filter's stop band is defined to begin at either the first local minimum in the magnitude response after the cutoff, or the first point of –60 dB attenuation, whichever frequency is lower. The three parameters shall be set as follows:

1. Band attenuation -60 dB;
2. Fn is 15% of Fs;
3. Maximum ripple 10% of the pass band value.

To be called an ISO/IEC 14496-3 audio decoder, the decoder shall provide an output satisfying the above conditions in the frequency domain for every filter (lopass, hipass, badpass, bandstop).

Testing of the bus width calculation and send/route mechanism shall be performed using test sequence SY014 and SY015. A reference output is provided by ISO/IEC as example. To be called an ISO/IEC 14496-3 audio decoder, the decoder shall provide two identical outputs for the two sequences, and in particular this output shall be composed of two channels characterized by a reverberated sound in the first and a dry sound in the second.

Testing of non-normative effects (**chorus**, **flange**, **reverb**) and the **spatialize** statement cannot be performed on objective constraints, since this functionality is implemented following many different and subjective criteria. As a consequence there are not any defined procedure to test this functionality. Content authors who wish to have normative effects processing such as reverberation should implement their own reverberation algorithms (for example) out of the strictly-normative building blocks and include them in the content as user-defined opcodes."

7.15.6 Descriptions of Conformance Bitstreams

All conformance bitstreams whose memory requirements and processing level, as indicated in Table 65 and Table 66, are less than or equal to that of a given level apply to that level.

Bitstream SY001 “math.mp4”

Math tests. Tests all “math” core opcode (5.9.4 in ISO/IEC 14496-3:2009 subpart 5). Produces a soundfile sampled at 20 Hz (not 20 kHz) for easy hand-checking. Heavy use of the **instr** statement.

Bitstream SY002 “buzz.mp4”

buzz test. Exercises **buzz** core opcode. Also uses **kline**, **cpsmidi**, **oscil**, **harm**, and a number of expressions.

Bitstream SY003 “pluck.mp4”

pluck test. Exercises **pluck** core opcode. Also uses **tableread, tablewrite, koscil, kline**, a number of expression types, and the **while** statement.

Bitstream SY004 “grain.mp4”

grain test. Exercises **grain** core opcode. Also uses **kexpon** and **expseg**.

Bitstream SY005 “piano.mp4”

Sampled piano. Uses bitstream samples, a bus, tablemaps, **fracdelay**, an opcode array, stereo output, and vector operations. Uses a MIDI file. Implements a complex, high-quality Gardner reverb. The decoded output of this bitstream is not sample-exact due to use of the **lopass()** core opcode.

Bitstream SY006 “bass.mp4”

Waveguide bass implementation. A complex algorithm integrating many functions. Uses core opcodes, filters, loops, tests, and tables heavily.

Bitstream SY007 “mixer.mp4”

Two simple sinusoidal inputs and a good quality two-channel mixer with low- and high-shelving functions and bell bandpass filters. Intense use of mathematical opcodes and iir filter. This bitstream is conceived expecially to test processing capabilities, it can easily be converted into an AudioFX orchestra; synthesis computation is minimal.

Bitstream SY008 “inmood.mp4”

Refrain of “In the mood”: a multiple instrument orchestra with tables without SASBF, FM, physical models and processing, several opcodes and table generators exercised. Complexity Level is Medium.

Bitstream SY009 “PC.mp4”

Complex synthesis, different instruments with peaks of very high polyphony. Highly demanding for floating-point operations, multiplications, mathematical methods *In some seconds, it does not fit in any of the defined Levels. This sequence is intended to stimulate implementers to design and to optimize advanced decoders, for complexity Levels that will be supported by future versions of the standard.*

Bitstream SY010 “sine1.mp4”

440 Hz sine, length 5 seconds + silence, length 1 second + 880 Hz sine, length 5 seconds + silence, length 1 second; sampling rate 32000 kHz, implemented with interp equal to 0 in the orchestra global block.

Bitstream SY011 “sine2.mp4”

440 Hz sine, length 5 seconds + silence, length 1 second + 880 Hz sine, length 5 seconds + silence, length 1 second; sampling rate 32000 kHz, implemented with interp equal to 1 in the orchestra global block.

Bitstream SY012 “noise.mp4”

Exercises the noise generators: sampling rate is 8 kHz. Each generator is active for 5 seconds, followed by 1 second of silence, in this order: linear distribution (-1,1), linearly-ramped distribution (0,1), exponential distribution (0.5), poissonian distribution (1/8000), gaussian (normal) distribution (0, 1). Note that the third and the fifth group contain saturated values, to 1 in the first case, to -1 and 1 in the second. This is already taken into account in the values given in 7.15.5 for test.

Bitstream SY013 “filters.mp4"

Exercises the lopass, hipass, bandpass, bandstop filters for SNR: sampling rate is 16 kHz. Each filter is active for 1 second, in the order described above, with white noise as input.

Sequence SY014 “clarinet1.mp4”

A clarinet synthesized at 44.1 kHz in FM with linear interpolation is routed to a reverberation instrument. The bus width is not set in the send statement, instead the sound is output twice to the bus, creating a two-channel bus. In the reverberation instrument only the first channel is reverberated, the second is output as is.

Sequence SY015 “clarinet2.mp4”

A clarinet synthesized at 44.1 kHz in FM with linear interpolation is routed to a reverberation instrument. The bus width is set to 2 in the send statement, the sound is output in mono to the bus, and then it must be replicated identical on the second channel. In the reverberation instrument only the first channel is reverberated, the second is output as is.

Sequence SY016 “fir.mp4”

In this sequence a sound synthesized at 32kHz is low pass filtered using both fir and firt core opcodes. The two filters have 16 coefficients and normalized cutoff frequency of 0.5 and 0.25 respectively. They are used to filter the left (fir) and right (firt) channels of the sound. This sequence also test pitch converters and other mathematical operators.

Sequence SY017 “vtone.mp4”

A sequence of monophonic sinusoidal tones with sampling rate frequency at 44.1 kHz is shaped according to attack and release time. This test sequence experiences mathematical operators, pitch converters and kline signal generator. Interpolation is linear.

Sequence SY018 “ttone.mp4”

A sequence of monophonic sinusoidal tones with sampling rate frequency at 44.1 kHz is shaped according to attack and release time. It is similar to SY017 but in this case tones are generated through tables instead that by mathematical functions. This test sequence experiences table generators, mathematical operators, table access and oscillators. Interpolation is linear.

Sequence SY019 “otone.mp4”

A sine tone is played in the left channel, with its octave tone in the right channel. Interpolation is linear, sampling rate is 44.1 kHz. This sequence especially experiences user defined core opcodes (including rate polymorphic), parameter passing, array variables.

Table 65 — Algorithmic Synthesis and Audio Fx Object Type Test Bitstreams

File Name	SY001	SY002	SY003	SY004	SY005	SY006	SY007
Content	math	buzz	pluck	grain	piano	bass	mixer
Processing Level	All	All	All	All	≥Med	High	All
RCU - RAM (KB)	< 4	< 4	< 4	< 4	3400	4	10

Table 66 — Algorithmic Synthesis and Audio Fx Object Type Test Bitstreams (continued)

File Name	SY008	SY009	SY010	SY011	SY012	SY013
Content	mood	PC	sin1	sin2	noise	filters
Processing level	≥Med	> *High*	All	All	All	All
RCU - RAM (KB)	3520	40	< 4	< 4	< 4	< 4

Table 67 — Algorithmic Synthesis and Audio Fx Object Type Test Sequences (continued)

File Name	SY014	SY015	SY016	SY017	SY018	SY019
Content	clarinet1	clarinet2	fir	vtone	ttone	otone
Processing level	All	All	≥Med	All	All	All
RCU - RAM (KB)	< 4	< 4	< 4	< 4	< 4	< 4

7.16 Main Synthetic

The main synthetic object type allows the use of all MPEG-4 Structured Audio tools (described in subpart 4 of the standard). It supports flexible, high-quality algorithmic synthesis using the SAOL music-synthesis language; efficient wavetable synthesis with the SASBF sample-bank format; and enables the use of high-quality mixing and postproduction in the Systems AudioBIFS toolset. Sound can be described at 0 kbps (no continuous cost) to 3-4 kbps for extremely expressive sounds in the MPEG-4 Structured Audio format.

There are four audio object types in Structured Audio: General MIDI, Wavetable Synthesis, Algorithmic Synthesis and Audio Fx, and Main Synthetic. Each of these object types corresponds to a particular set of application requirements. The default object type is the Main Synthetic Object type; when reference is made to MPEG-4 Structured Audio format without reference to a object type, it shall be understood that the reference is to the Main Synthetic Object type.

7.16.1 DecoderSpecificInfo Characteristics

Bitstream provider may apply restrictions to the following parameters of the DecoderSpecificInfo:

Any restrictions specified by the MIDI, Wavetable synthesis and Algorithmic synthesis and AudioFX apply.

7.16.2 Audio Access Unit Characteristics

Bitstream provider may apply restrictions to the following parameters of the Access Units:

Any restrictions specified by the MIDI, Wavetable synthesis and Algorithmic synthesis and AudioFX apply.

7.16.3 Procedure to Test Bitstream Conformance

Bitstreams for the main synthetic profile must conform to the description in ISO/IEC 14496-3 subpart 4 in both syntax and complexity. Any other restrictions specified by the MIDI, Wavetable synthesis and Algorithmic synthesis and AudioFX apply.

7.16.3.1 DecoderSpecificInfo Characteristics

AudioObjectType: Shall be encoded with the value 13

SamplingFrequencyIndex: Shall be encoded with the value 0 to 0xc

SamplingFrequency: Shall be encoded with the value 0 to 96000.

ChannelConfiguration: Shall be encoded with the value 0 to 7.

The following restrictions apply to StructuredAudioSpecificConfig:

Any restrictions specified by the MIDI, Wavetable synthesis and Algorithmic synthesis and AudioFX apply.

7.16.3.2 Audio Access Unit Characteristics

Any restrictions specified by the MIDI, Wavetable synthesis and Algorithmic synthesis and AudioFX apply.

7.16.4 Procedure to Test Decoder Conformance

All profiles that support the Main Synthetic audio object type must conform to the procedures specified for the following audio object types:

- MIDI
- Wavetable synthesis
- Algorithmic synthesis and AudioFX

7.16.5 Descriptions of Conformance Bitstreams

See sections on the following audio object types:

- MIDI
- Wavetable synthesis
- Algorithmic synthesis and AudioFX

7.17 SBR

7.17.1 Compressed data

7.17.1.1 Characteristics

For all applicable Audio Object Types the SBR extension_payload() elements should be placed last among the extension_payload() elements, i.e. if another type of extension_payload() element is present it should be placed prior to the SBR extension_payload() elements.

If the Audio Object Type SBR is used in combination with either of the Audio Object Types AAC main, AAC LC, AC SSR or AAC LTP, the compressed data shall be stored as outlined in ISO/IEC 14496-3:2009, 4.5.2.8.2.2: SBR Extension Payload for the Audio Object Types AAC main, AAC SSR, AAC LC and AAC LTP.

If the Audio Object Type SBR is used in combination with either of the Audio Object Types ER AAC LC or ER AAC LTP, the compressed data shall be stored as outlined in ISO/IEC 14496-3:2009, 4.5.2.8.2.3: SBR Extension Payload for the Audio Object Types ER AAC LC and ER AAC LTP. For these AOTs, DRC extension_payload() elements are not permitted simultaneously with SBR extension_payload() elements within one er_raw_data_block(). Moreover, SBR extension_payload() elements of the type EXT_SBR_DATA_CRC shall not be used with these AOTs.

For the scalable AOTs (AAC scalable and ER AAC scalable), the SBR data should be transmitted and devised according to ISO/IEC 14496-3:2009, 4.5.2.8.2.4 SBR extension payload for the Audio Object Types AAC scalable and ER AAC scalable. Restrictions are here put on the frequency range of the SBR data and in what layers of the scalable stream the SBR data is stored. Furthermore, SBR extension_payload() elements of the type EXT_SBR_DATA_CRC shall not be used with the audio object types ER AAC scalable.

7.17.1.2 Test procedure

Each compressed data shall meet the syntactic and semantic requirements specified in ISO/IEC 14496-3. The decoded data shall also meet the requirements defined in ISO/IEC 14496-3:2009, 4.6.18.3.6 Requirements. If a syntactic element is not listed below, no restrictions apply to that element. The **bs_reserved** elements shall be encoded with the value zero.

7.17.1.2.1 Compressed MPEG-4 data payload

7.17.1.2.1.1 sbr_header()

The following parameters shall be encoded with values subsequently used in defining a frequency range, a number of noise bands, a number of limiter bands, and a number of patches:

bs_start_freq
bs_stop_freq
bs_xover_band
bs_alter_scale
bs_noise_bands
bs_limiter_bands

The above parameters are used (in ISO/IEC 14496-3) to calculate the variables below:

k_2
k_0
k_x
M
N_Q
numPatches
numBands
numBands0

vDk0
vDk1

Conformant compressed MPEG-4 data shall have values for the above parameters that subsequently evaluate to values of the above variables that satisfy the requirements outlined in ISO/IEC 14496-3:2009, 4.6.18.3.6 Requirements.

7.17.1.2.1.2 sbr_channel_pair_base_element()

bs_coupling: Shall be encoded with the value of 1

7.17.1.2.1.3 sbr_grid()

The following compressed MPEG-4 data elements shall be encoded so that a value of the number of SBR envelopes for a SBR frame, for a given frame class, is within the limits defined in ISO/IEC 14496-3:2009, 4.6.18.3.6 Requirements:

bs_rel_bord_0
bs_rel_bord_1
bs_num_env
bs_var_bord
bs_num_rel_0
bs_var_bord_0
bs_var_bord_1
bs_num_rel_0
bs_num_rel_1

Conformant compressed MPEG-4 data shall have the above parameters chosen so that the leading border of a given SBR frame (the frame boundary) coincides with the trailing border of the previous SBR frame (the frame boundary of the previous frame). Furthermore, the above parameters shall be chosen so that the envelope borders of the SBR envelopes in a given frame fall within the boundaries of the SBR frame. The above parameters shall also be chosen so that every SBR envelope within the SBR frame has a duration larger than zero.

7.17.1.2.1.4 sbr_dtdf()

bs_df_env[]: Shall be encoded with the value 0 for the first envelope of the present frame, if the compressed MPEG-4 data element bs_header_flag has the value one (i.e. a new sbr_header is available), or if the amp_res value has changed from the previous frame due to the rule specifying amp_res = 0 for a frame of frame class FIXFIX with only one envelope.

bs_df_noise[]: Shall be encoded with the value 0 for the first noise floor of the present frame, if the compressed MPEG-4 data element bs_header_flag has the value one, i.e. a new sbr_header is available.

7.17.1.2.1.5 sbr_envelope()

bs_codeword: Shall be encoded with the values listed in the corresponding Huffman table, defined in ISO/IEC 14496-3, Annex 4.A.6.1

Conformant compressed MPEG-4 data shall have coded envelope scalefactors based on quantized envelopes scalefactors that satisfy the requirements outlined in ISO/IEC 14496-3:2009, 4.6.18.3.6 Requirements.

The quantised envelope scale factors $\mathbf{E}$ for single channel elements and $\mathbf{E}_0$ and $\mathbf{E}_1$ for channel pair elements shall be encoded with values that are within the following limits:

- For single channel elements: $0 \le \mathbf{E}(i,l) < 2^{7-amp\_res}$
- For channel pair elements: $\begin{cases} 0 \le \mathbf{E}_0(i,l) < 2^{7-amp\_res} \\ 0 \le \mathbf{E}_1(i,l) < 2^{7-amp\_res-bs\_coupling} \end{cases}$

where

$$amp\_res = \begin{cases} 0 & ,\text{if } bs\_num\_env = 1 \text{ and } frame\_class = FIXFIX \\ bs\_amp\_res & ,otherwise \end{cases}$$

where subscript zero indicates the firstly encoded channel in the channel pair element and subscript one indicates the secondly encoded channel in the channel pair element.

7.17.1.2.1.6 sbr_noise()

bs_codeword: Shall be encoded with the values listed in the corresponding Huffman table, defined in ISO/IEC 14496-3, Annex 4.A.6.1

Conformant compressed MPEG-4 data shall have coded noise floor scalefactors based on quantised noise floor scalefactors that satisfy the requirements outlined in ISO/IEC 14496-3:2009, 4.6.18.3.6 Requirements.

7.17.2 Decoders

7.17.2.1 Characteristics

The object type SBR has the Object Type ID 5, and the compressed MPEG-4 data syntax is defined in ISO/IEC 14496-3. The Audio Object Type SBR contains the SBR Tool. The SBR Tool can be implemented in two different versions:

- High-Quality SBR Tool
- Low-Power SBR Tool

The different versions can also be operated in down-sampled SBR-mode.

The internal sampling rate of the SBR Tool shall always be twice the sampling rate indicated by samplingFrequency or samplingFrequencyIndex in the AudioSpecificConfig().

A conformant implementation of the SBR tool that receives an SBR enhanced data stream shall operate in upsampling mode only, until an sbr_header is received, ensuring that the SBR data can be decoded correctly.

7.17.2.1.1 HE-AAC

The ability to do down-sampled SBR is mandatory for levels 3 and 4 of the High Efficiency AAC Profile.

A decoder conforming to that profile and level shall operate the down-sampled SBR tool if one of the following conditions is fulfilled:

- extensionSamplingFrequency is the same as samplingFrequency, or
- extensionSamplingFrequencyIndex is the same as samplingFrequencyIndex, or

the output sampling rate would otherwise exceed the maximum allowed output sample rate for the given level.

An HE-AAC profile decoder shall support implicit SBR signaling, as outlined in ISO/IEC 14496-3:2009, 1.6.5.3 HE AAC Profile Decoder Behavior in Case of Implicit Signaling.

An HE-AAC profile decoder shall support explicit SBR signaling as outlined in ISO/IEC 14496-3:2009, 1.6.5.4 HE AAC Profile Decoder Behavior in Case of Explicit Signaling.

7.17.2.2 SBR conformance test procedure

For the sake of simplicity, conformance testing of the AOT SBR is carried out in conjunction with AAC LC.

The conformance test procedure for the SBR tool internally creates a reference for comparison, given an input compressed MPEG-4 data file and the output from the decoder under test. In order to accomplish this, apart from the *_twi_* and *_qmf_* types, every SBR conformance compressed MPEG-4 data file is divided into two parts as outlined in Figure 10, where the AAC data for the two parts is identical but the SBR header does not arrive until the second part. This ensures that in the case of implicit signaling a conformant decoder will recognise the SBR extension element at the beginning of the compressed MPEG-4 data, or in the case of explicit signalling, by parsing the audioSpecificConfig(). Since no SBR header is present it cannot start SBR decoding and will hence do up-sampling using the SBR QMF filterbank in anticipation of the SBR header.

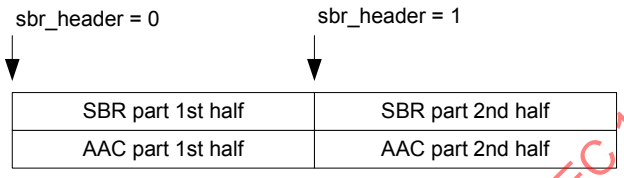

Figure 10 — The disposition of the SBR conformance compressed MPEG-4 data files

The conformance test procedure stipulate:

- reading the compressed MPEG-4 data file;
- while no SBR header is present taking the input decoded file (the output from the decoder under test) and down-sample the signal;
- store the down-sampled signal.

Since this signal is just an up-sampled version of the output-signal from the AAC and hence the input signal to the SBR decoder under test, it can be down-sampled, and by means of a polyphase correction filter, be approximated to be the same signal as was used by the SBR Tool in the decoder under test.

In parallel to storing the signal it shall also be fed to the reference SBR decoder where, since no SBR header is available, up-sampling is performed. The, in the reference SBR decoder, upsampled signal shall be compared to the input signal, i.e. the output from the decoder under test. This serves as a QMF test of the first half of the conformance file.

When the SBR header arrives, a SBR processed reference signal based on the stored lowband signal (that is a very close approximation of the signal that the SBR Tool in the decoder under test used) shall be calculated.

This means that it is possible to test the accuracy of the SBR part of the implementation, without having to deal with the differences between the AAC implementation used in the decoder under test, and the AAC implementation used for producing reference waveforms. Furthermore, the accuracy of the QMF implementation of the decoder under test is tested separately for every conformance sequence.

If a complex QMF filter bank with a modified internal phase angle (hereinafter referred to as twiddles) is used in the decoder under test, the al_sbr_twi_* sequences shall be tested first. Contrary to the other al_sbr_* test-sequences, these sequences consist of zero signal AAC-data and SBR elements that trigger sine-addition in the entire frequency range covered by SBR. The output-signal of this test sequence will contain sinusoids with phase angles that depend on the implementation of the synthesis filterbank in the decoder under test. Based on the obtained output signal from the decoder under test, the two parameters φ and β describing the phase characteristics of the filter banks in the decoder under test are identified. Since different phase characteristics may be used, depending on whether downsampled SBR is used or not, both al_sbr_twi_* sequences must be run prior to conformance testing. The file sequence is only used to perform an examination and identification of the filter bank. It does not test conformance of the QMF bank. The performance of the total QMF (analysis – synthesis) shall be tested with the *_qmf_* sequences and the other informative QMF bank tests.

In order to ensure that the QMF is implemented correctly, the output from the QMF test specific sequences *_qmf_* are compared to the internal reference without down-sampling and storing the first half of the test sequence. This is since it is possible, however very unlikely, to introduce errors in the QMF implementation that from the SBR conformance test procedure point of view look like differences in the AAC implementation. By, for the QMF specific sequences, omitting the parts of the tool that are designed to neglect differences in the AAC implementation it is ensured that the QMF is implemented correctly.

If the decoder under test passes the conformance criteria for the dedicated QMF test sequences, this is a good indication that the QMF implementation is accurate. However, it is no definite guarantee, and hence it could happen that a QMF implementation that barely passes the conformance for the QMF test, does not pass conformance for other parts of the system due to the QMF implementation. Therefore, it is useful to observe the result from the QMF test for the first half for any of the conformance sequences. This can give a good indication of the origin of a potential error.

Figure 11 outlines the SBR conformance test procedure.

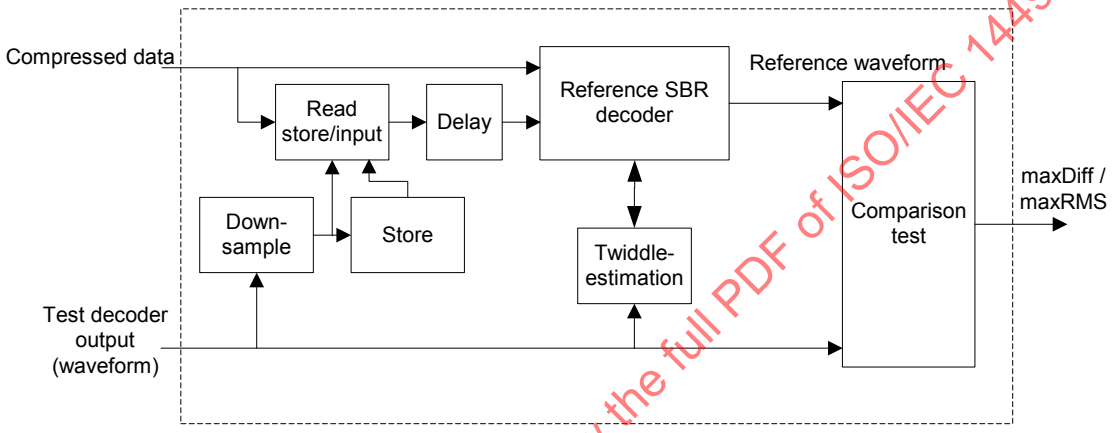

Figure 11 — Block diagram of the SBR conformance test procedure

The essential modules are:

- **Read store/input**: This module parses the SBR part of the compressed MPEG-4 data and searches for the SBR header. When no SBR header is available, the downsampled Test decoder output is routed to both the Storage Module and through the Delay module to the Reference SBR decoder. When an SBR header is available, the data stored in the Storage module is routed through the Delay module to the Reference SBR decoder. The Test decoder output is routed to the comparison test module.

- **Down-sample**: This module downsamples the Test decoder output signal, by decimation, and applies a polyphase filter that approximates the inverse of the equivalent polyphase filter of the QMF-upsampler. The delay imposed by the downsampler is given by:

  $delay = 32 \cdot \left(\frac{K-1}{2} \right)$, where $K = 25$ is the length of the polyphase filter. If down-sampled SBR is used, the down-sampler omits the decimation and does only the polyphase filtering. The polyphase filter matrix $\mathbf{H}(k,l)$ of size $32 \times K$ is tabulated in Table 68. The polyphase filtering step consists of the operation which maps a time signal $x(n)$ to $y(n)$, where

  $$y(k+32i) = \sum_{l=0}^{K-1} \mathbf{H}(k,l)\, x\big(k+32(i-l)\big), \quad k = 0,1,\ldots,31.$$

- **Twiddle-estimation:** This module identifies the twiddles used in the synthesis filter bank in the codec under test, and based on this computes the analysis filterbank twiddles in the codec under test. This information is passed on to the reference SBR decoder where the QMF filter bank is given the same distribution of the twiddle factors as used in the decoder under test.

- **Reference SBR decoder** is a reference SBR decoder according to the ISO specification. It generates a reference signal based on the stored low-band signal and the compressed MPEG-4 SBR data. The delay of the reference SBR decoder (at the input sampling rate) is 481 samples.

- **Comparison test,** this module calculates the difference signals between the output from the decoder under test and the internal reference. The maximum amplitude of the difference signal as well as the RMS of the difference signal are calculated. These conformance criteria are specified with respect to PCM-sample in the range –32768 .. 32767.

The process of identifying the QMF filter bank is outlined in the following steps:

- Decode the SBR compressed data so that the SBR start band k_x and number of SBR bands M are determined.

- Set the following variables:

  Upsampling factor $upS = 2 - bDownSampledSbr$,
  DFT analysis length $fftLen = 16384\, upS$,
  Number of synthesis QMF channels $L = 32\, upS$,
  Prototype filter order $N = 320\, upS$, and
  SBR stop band $k_y = \min(k_x + M, L)$,
  where *bDownSampledSbr* is 1 if down-sampled SBR is used, otherwise zero.

- Read at least $fftLen$ samples from the mono/left channel output of the decoder under test and store them in the array $inputSignal(n)$.

- Compute DFT with frequency offset $H(k)$ as

$$H(k) = \sum_{n=0}^{fftLen-1} inputSignal(n) \exp\left(-i\frac{\pi}{L}(k+0.5)n\right), \quad k_x \le k < k_y$$

- Compute the scaled unwrapped phase $P(k)$ according to the following pseudo code:

```
P(k_x-1) = 0
for ( k = k_x; k < k_y; k++) {
    P(k) = ATAN2( IMAG(H(k)), REAL(H(k)) ) / π
    if(P(k) > P(k-1))
        while(P(k) - P(k-1) > 1)
            P(k) = P(k) - 2
    else
        while(P(k-1) - P(k) > 1)
            P(k) = P(k) + 2
}
```

  where $ATAN2(\ )$ is the arc tangent in four quadrants, and $REAL(\ )$ and $IMAG(\ )$ are operators to extract the real and the imaginary parts of the argument respectively.

- Perform linear curve fitting using least square error minimization as

$$K = \sum_{k=k_x}^{k_y-1} k^2 \qquad R = \sum_{k=k_x}^{k_y-1} k$$

$$S = \sum_{k=k_x}^{k_y-1} k\,P(k) \quad T = \sum_{k=k_x}^{k_y-1} P(k)$$

$$f(k) = ak + b, \quad where$$

$$a = \frac{(k_y - k_x)\,S - R\,T}{K\,(k_y - k_x) - R^2} \quad and \quad b = \frac{S - K\,a}{R}$$

- Identification of the terms of the complex exponent

  $-j\,\pi\,f(k) = -j\,\pi\,(ak + b)$ with

  $-j\frac{\pi}{L}\{(k+0.5)\varphi + \beta\}$ makes

  $\varphi = L\,a$

  $\beta = L\,b - \frac{\varphi}{2}$

- Quantize φ in $1/L$ steps and β in steps of one as

  $$\varphi = \frac{NINT(L\,\varphi)}{L}$$
  $$\beta = NINT(\beta)$$

- The synthesis QMF bank can subsequently be identified as using the modulation matrix $N(k,n)$ as

  $$N(k,n) = \frac{1}{64}\exp\left(i\frac{\pi}{L}\{(k+0.5)\varphi + \beta\}\right)\exp\left(i\frac{\pi}{L}(k+0.5)n\right) =$$
  $$= \frac{1}{64}\exp\left(i\frac{\pi}{L}\{(k+0.5)(n+\varphi) + \beta\}\right) \quad ,\begin{cases} 0 \le k < L \\ 0 \le n < 2L \end{cases}$$

- From the synthesis modulation expression, the analysis modulation matrix $M(k,n)$ may be obtained as

  $$M(k,n) = \exp\left(-i\frac{\pi}{L}\{(k+0.5)(N+\varphi) + \beta\}\right)\exp\left(i\frac{\pi}{L_A}(k+0.5)n\right) =$$
  $$= \exp\left(i\frac{\pi}{L_A}\left\{(k+0.5)\left(n - N_A - \frac{\varphi}{upS}\right) - \frac{\beta}{upS}\right\}\right) \quad ,\begin{cases} 0 \le k < L_A \\ 0 \le n < 2L_A \end{cases}$$

  where

  L_A is the number of analysis QMF channels (32), and
  N_A is the prototype filter order for the analysis QMF bank (320).

The above calculated analysis and synthesis matrices should be used when calculating the reference SBR decoded signal that is to be compared with the output signal of the decoder under test.

Table 68 — Filter coefficients for the polyphase downsample filter

```
H[32][25] = {
   {3.476697953131265e-008, 0, -6.763157773587183e-008, 0, -6.974973600673098e-006, 0,
   -1.924219571106937e-004, 0,  1.629822651831791e-004, 0,  3.540528761884519e-004, 0,
    9.999940276586996e-001, 0,  3.540528761884519e-004, 0,  1.629822651831791e-004, 0,
   -1.924219571106937e-004, 0, -6.974973600673098e-006, 0, -6.763157773587183e-008, 0,
    3.476697953131265e-008},
   {1.396471916031115e-007, 0, -2.974245852183672e-007, 0, -1.318138373054617e-005, 0,
   -3.865443699577471e-004, 0,  3.721410827846135e-004, 0,  3.666791540864976e-004, 0,
    9.999958594813760e-001, 0,  3.666791540864976e-004, 0,  3.721410827846135e-004, 0,
   -3.865443699577471e-004, 0, -1.318138373054617e-005, 0, -2.974245852183672e-007, 0,
    1.396471916031115e-007},
   {1.224963549817200e-007, 0, -2.646404681011980e-007, 0, -1.108178601327391e-005, 0,
   -3.610222352290030e-004, 0,  3.554111749102114e-004, 0,  3.596349479868729e-004, 0,
    1.000005611329237e+000, 0,  3.596349479868729e-004, 0,  3.554111749102114e-004, 0,
   -3.610222352290030e-004, 0, -1.108178601327391e-005, 0, -2.646404681011980e-007, 0,
    1.224963549817200e-007},
   {1.326399319916421e-007, 0, -2.700352437567778e-007, 0, -9.882252437042349e-006, 0,
   -3.735656036780282e-004, 0,  3.517646637852679e-004, 0,  3.633285653067748e-004, 0,
    1.000010164947417e+000, 0,  3.633285653067748e-004, 0,  3.517646637852679e-004, 0,
   -3.735656036780282e-004, 0, -9.882252437042349e-006, 0, -2.700352437567778e-007, 0,
    1.326399319916421e-007},
   {1.522562232650754e-007, 0, -2.984765811933622e-007, 0, -8.632246089961839e-006, 0,
   -3.981722856143535e-004, 0,  3.668277251617622e-004, 0,  3.652031799326229e-004, 0,
    1.000002336408856e+000, 0,  3.652031799326229e-004, 0,  3.668277251617622e-004, 0,
   -3.981722856143535e-004, 0, -8.632246089961839e-006, 0, -2.984765811933622e-007, 0,
    1.522562232650754e-007},
   {1.770077632287240e-007, 0, -3.191765536892460e-007, 0, -7.252506589625763e-006, 0,
   -4.269398602904086e-004, 0,  3.673910838620092e-004, 0,  3.739888363247150e-004, 0,
    1.000002916620203e+000, 0,  3.739888363247150e-004, 0,  3.673910838620092e-004, 0,
   -4.269398602904086e-004, 0, -7.252506589625763e-006, 0, -3.191765536892460e-007, 0,
    1.770077632287240e-007},
   {2.068765829715687e-007, 0, -3.511660788410439e-007, 0, -5.652904186405254e-006, 0,
   -4.594357796680696e-004, 0,  3.776142052929487e-004, 0,  3.659398368134911e-004, 0,
    1.000009380633999e+000, 0,  3.659398368134911e-004, 0,  3.776142052929487e-004, 0,
   -4.594357796680696e-004, 0, -5.652904186405254e-006, 0, -3.511660788410439e-007, 0,
    2.068765829715687e-007},
   {2.421414005440356e-007, 0, -3.782805603719183e-007, 0, -3.994773586637473e-006, 0,
   -4.950706931314535e-004, 0,  3.790387553154356e-004, 0,  3.671113185055310e-004, 0,
    1.000014455098329e+000, 0,  3.671113185055310e-004, 0,  3.790387553154356e-004, 0,
   -4.950706931314535e-004, 0, -3.994773586637473e-006, 0, -3.782805603719183e-007, 0,
    2.421414005440356e-007},
   {2.788460966259352e-007, 0, -4.079961401854086e-007, 0, -2.260958885598900e-006, 0,
   -5.296088142178838e-004, 0,  3.835318699545681e-004, 0,  3.763419839421675e-004, 0,
    9.999940885015929e-001, 0,  3.763419839421675e-004, 0,  3.835318699545681e-004, 0,
   -5.296088142178838e-004, 0, -2.260958885598900e-006, 0, -4.079961401854086e-007, 0,
    2.788460966259352e-007},
   {3.161911779625002e-007, 0, -4.313203422567221e-007, 0, -5.273303865290991e-007, 0,
   -5.625765678906092e-004, 0,  3.829529676084116e-004, 0,  3.673391073119114e-004, 0,
    9.999951684002906e-001, 0,  3.673391073119114e-004, 0,  3.829529676084116e-004, 0,
   -5.625765678906092e-004, 0, -5.273303865290991e-007, 0, -4.313203422567221e-007, 0,
    3.161911779625002e-007},
   {3.543122532697675e-007, 0, -4.608625488256222e-007, 0,  1.184078570054111e-006, 0,
   -5.943681506843667e-004, 0,  3.884059372906167e-004, 0,  3.792552922175888e-004, 0,
    9.999972692390483e-001, 0,  3.792552922175888e-004, 0,  3.884059372906167e-004, 0,
   -5.943681506843667e-004, 0,  1.184078570054111e-006, 0, -4.608625488256222e-007, 0,
    3.543122532697675e-007},
   {3.909402775633086e-007, 0, -4.739340938681313e-007, 0,  2.706078490008691e-006, 0,
   -6.234965573461278e-004, 0,  3.816668617870812e-004, 0,  3.772857914767589e-004, 0,
    1.000002886171420e+000, 0,  3.772857914767589e-004, 0,  3.816668617870812e-004, 0,
   -6.234965573461278e-004, 0,  2.706078490008691e-006, 0, -4.739340938681313e-007, 0,
    3.909402775633086e-007},
```

```
{4.222809367831911e-007, 0, -4.878510064125678e-007, 0,  4.036795191477909e-006, 0,
-6.473621895606138e-004, 0,  3.791705819779010e-004, 0,  3.847083857331159e-004, 0,
 9.999969286735556e-001, 0,  3.847083857331159e-004, 0,  3.791705819779010e-004, 0,
-6.473621895606138e-004, 0,  4.036795191477909e-006, 0, -4.878510064125678e-007, 0,
 4.222809367831911e-007},
{4.502138847524999e-007, 0, -4.999326042535363e-007, 0,  5.135058417623972e-006, 0,
-6.679743577822745e-004, 0,  3.771635262404024e-004, 0,  3.866181819880556e-004, 0,
 9.999972973017242e-001, 0,  3.866181819880556e-004, 0,  3.771635262404024e-004, 0,
-6.679743577822745e-004, 0,  5.135058417623972e-006, 0, -4.999326042535363e-007, 0,
 4.502138847524999e-007},
{4.709334223508883e-007, 0, -5.078830669590950e-007, 0,  5.962080730857686e-006, 0,
-6.828692913148293e-004, 0,  3.751956167593989e-004, 0,  3.827003698106720e-004, 0,
 1.000008336100520e+000, 0,  3.827003698106720e-004, 0,  3.751956167593989e-004, 0,
-6.828692913148293e-004, 0,  5.962080730857686e-006, 0, -5.078830669590950e-007, 0,
 4.709334223508883e-007},
{4.843159761574267e-007, 0, -5.062025248855524e-007, 0,  6.458113652318982e-006, 0,
-6.923844198753944e-004, 0,  3.691031022061020e-004, 0,  3.826743784427686e-004, 0,
 1.000007446899661e+000, 0,  3.826743784427686e-004, 0,  3.691031022061020e-004, 0,
-6.923844198753944e-004, 0,  6.458113652318982e-006, 0, -5.062025248855524e-007, 0,
 4.843159761574267e-007},
{4.881790944646534e-007, 0, -5.091494784777248e-007, 0,  6.616136460265565e-006, 0,
-6.950689516769845e-004, 0,  3.698758977096631e-004, 0,  3.826493196713532e-004, 0,
 9.999860218829778e-001, 0,  3.826493196713532e-004, 0,  3.698758977096631e-004, 0,
-6.950689516769845e-004, 0,  6.616136460265565e-006, 0, -5.091494784777248e-007, 0,
 4.881790944646534e-007},
{4.843159761574265e-007, 0, -5.062025248858790e-007, 0,  6.458113652319081e-006, 0,
-6.923844198753948e-004, 0,  3.691031022060960e-004, 0,  3.826743784426941e-004, 0,
 1.000007446899662e+000, 0,  3.826743784426941e-004, 0,  3.691031022060960e-004, 0,
-6.923844198753948e-004, 0,  6.458113652319081e-006, 0, -5.062025248858790e-007, 0,
 4.843159761574265e-007},
{4.709334223509752e-007, 0, -5.078830668471703e-007, 0,  5.962080730857398e-006, 0,
-6.828692913148301e-004, 0,  3.751956167594234e-004, 0,  3.827003698109648e-004, 0,
 1.000008336100521e+000, 0,  3.827003698109648e-004, 0,  3.751956167594234e-004, 0,
-6.828692913148301e-004, 0,  5.962080730857398e-006, 0, -5.078830668471703e-007, 0,
 4.709334223509752e-007},
{4.502138847525010e-007, 0, -4.999326042533778e-007, 0,  5.135058417624179e-006, 0,
-6.679743577822758e-004, 0,  3.771635262403889e-004, 0,  3.866181819878970e-004, 0,
 9.999972973017255e-001, 0,  3.866181819878970e-004, 0,  3.771635262403889e-004, 0,
-6.679743577822758e-004, 0,  5.135058417624179e-006, 0, -4.999326042533778e-007, 0,
 4.502138847525010e-007},
{4.222809367831921e-007, 0, -4.878510064126016e-007, 0,  4.036795191477679e-006, 0,
-6.473621895606155e-004, 0,  3.791705819779300e-004, 0,  3.847083857334735e-004, 0,
 9.999969286735594e-001, 0,  3.847083857334735e-004, 0,  3.791705819779300e-004, 0,
-6.473621895606155e-004, 0,  4.036795191477679e-006, 0, -4.878510064126016e-007, 0,
 4.222809367831921e-007},
{3.909402775633092e-007, 0, -4.739340938680947e-007, 0,  2.706078490009059e-006, 0,
-6.234965573461287e-004, 0,  3.816668617870499e-004, 0,  3.772857914763712e-004, 0,
 1.000002886171421e+000, 0,  3.772857914763712e-004, 0,  3.816668617870499e-004, 0,
-6.234965573461287e-004, 0,  2.706078490009059e-006, 0, -4.739340938680947e-007, 0,
 3.909402775633092e-007},
{3.543122532697675e-007, 0, -4.608625488256092e-007, 0,  1.184078570054243e-006, 0,
-5.943681506843663e-004, 0,  3.884059372906052e-004, 0,  3.792552922174683e-004, 0,
 9.999972692390470e-001, 0,  3.792552922174683e-004, 0,  3.884059372906052e-004, 0,
-5.943681506843663e-004, 0,  1.184078570054243e-006, 0, -4.608625488256092e-007, 0,
 3.543122532697675e-007},
{3.161911779624999e-007, 0, -4.313203422567522e-007, 0, -5.273303865294305e-007, 0,
-5.625765678906092e-004, 0,  3.829529676084380e-004, 0,  3.673391073122771e-004, 0,
 9.999951684002906e-001, 0,  3.673391073122771e-004, 0,  3.829529676084380e-004, 0,
-5.625765678906092e-004, 0, -5.273303865294305e-007, 0, -4.313203422567522e-007, 0,
 3.161911779624999e-007},
{2.788460966259361e-007, 0, -4.079961401853398e-007, 0, -2.260958885598140e-006, 0,
-5.296088142178834e-004, 0,  3.835318699545062e-004, 0,  3.763419839414048e-004, 0,
 9.999940885015904e-001, 0,  3.763419839414048e-004, 0,  3.835318699545062e-004, 0,
-5.296088142178834e-004, 0, -2.260958885598140e-006, 0, -4.079961401853398e-007, 0,
 2.788460966259361e-007},
```

```
    {2.421414005440362e-007, 0, -3.782805603718790e-007, 0, -3.994773586636954e-006, 0,
    -4.950706931314529e-004, 0,  3.790387553153991e-004, 0,  3.671113185050744e-004, 0,
     1.000014455098326e+000, 0,  3.671113185050744e-004, 0,  3.790387553153991e-004, 0,
    -4.950706931314529e-004, 0, -3.994773586636954e-006, 0, -3.782805603718790e-007, 0,
     2.421414005440362e-007},
    {2.068765829715683e-007, 0, -3.511660788410130e-007, 0, -5.652904186404924e-006, 0,
    -4.594357796680679e-004, 0,  3.776142052929195e-004, 0,  3.659398368131350e-004, 0,
     1.000093806339960e+000, 0,  3.659398368131350e-004, 0,  3.776142052929195e-004, 0,
    -4.594357796680679e-004, 0, -5.652904186404924e-006, 0, -3.511660788410130e-007, 0,
     2.068765829715683e-007},
    {1.770077632288873e-007, 0, -3.191765536890821e-007, 0, -7.252506589403714e-006, 0,
    -4.269398602904083e-004, 0,  3.673910838620111e-004, 0,  3.739888363247353e-004, 0,
     1.000002916620202e+000, 0,  3.739888363247353e-004, 0,  3.673910838620111e-004, 0,
    -4.269398602904083e-004, 0, -7.252506589403714e-006, 0, -3.191765536890821e-007, 0,
     1.770077632288873e-007},
    {1.522562232650743e-007, 0, -2.984765811933783e-007, 0, -8.632246089962066e-006, 0,
    -3.981722856143523e-004, 0,  3.668277251617779e-004, 0,  3.652031799328445e-004, 0,
     1.000002336408854e+000, 0,  3.652031799328445e-004, 0,  3.668277251617779e-004, 0,
    -3.981722856143523e-004, 0, -8.632246089962066e-006, 0, -2.984765811933783e-007, 0,
     1.522562232650743e-007},
    {1.326399319916421e-007, 0, -2.700352437567779e-007, 0, -9.882252437042350e-006, 0,
    -3.735656036780283e-004, 0,  3.517646637852678e-004, 0,  3.633285653067912e-004, 0,
     1.000010164947417e+000, 0,  3.633285653067912e-004, 0,  3.517646637852678e-004, 0,
    -3.735656036780283e-004, 0, -9.882252437042350e-006, 0, -2.700352437567779e-007, 0,
     1.326399319916421e-007},
    {1.224963549817213e-007, 0, -2.646404681008590e-007, 0, -1.108178601327352e-005, 0,
    -3.610222352290024e-004, 0,  3.554111749101667e-004, 0,  3.596349479863394e-004, 0,
     1.000005611329234e+000, 0,  3.596349479863394e-004, 0,  3.554111749101667e-004, 0,
    -3.610222352290024e-004, 0, -1.108178601327352e-005, 0, -2.646404681008590e-007, 0,
     1.224963549817213e-007},
    {1.396471916031121e-007, 0, -2.974245852182784e-007, 0, -1.318138373054608e-005, 0,
    -3.865443699577474e-004, 0,  3.721410827846046e-004, 0,  3.666791540863757e-004, 0,
     9.999958594813758e-001, 0,  3.666791540863757e-004, 0,  3.721410827846046e-004, 0,
    -3.865443699577474e-004, 0, -1.318138373054608e-005, 0, -2.974245852182784e-007, 0,
     1.396471916031121e-007},
};
```

The test procedures specified in Table 69 have to be applied. The testing shall be done using the test procedure defined above. The conformance of the underlying AAC decoder shall be tested before conformance testing is done for the SBR Tool. The SBR Tool is based on a pseudo QMF filterbank. This is the most critical part of the SBR Tool in terms of precision. Hence, in order to simplify conformance testing, it is recommended to first check the accuracy of the QMF implementation, after proper identification of the filterbank twiddling in the decoder under test.

The Low Power SBR Tool has an aliasing detection algorithm that includes threshold binary decisions which can give rise to large output differences due to small rounding errors. In the aliasing detection algorithm a decision is made on how to modify gain calculation in the envelope adjuster. If potentially strong aliasing is detected, a clear indication on how to modify the gain values will be obtained, and no rounding problems will occur. However, if no strong aliasing is detected, the gain values will be modified anyway, albeit to a much smaller extent. Since no strong aliasing is detected, the modification decision could vary due to rounding effects in the aliasing detection algorithm. A small rounding error will then cause the gain calculation algorithm to modify different gain values, and hence an inaudible, but clearly measurable difference is observed in the output. The conformance sequences are designed to avoid the above problem, but differences can appear for other sequences.

The inverse filtering in the HF generator is another module that is prone to give output differences due to rounding effects. Similarly to the differences due to rounding in the aliasing detection, these are inaudible differences.

Reference waveform output files from decoding the compressed MPEG-4 data conformance sequences with the MPEG-4 reference software are also available. These files are available for reference only, and are not part of the official conformance testing procedure.

Notes: The reference software implementation of the conformance test procedure will behave differently for the test sequences named al_sbr_twi_*, since for these test sequences the QMF implementation is identified with respect to the twiddle factors used on the analysis and synthesis side of the QMF.

The reference software implementation of the conformance test procedure will behave differently for the test sequences named al_sbr_qmf_*, since for these test sequences only the QMF implementation is tested, as outlined above in the tool description. For these sequences only the output from the QMF test is displayed. This QMF test is not informative as for the other sequences, it is normative and shall be passed in order for the device under test to be conformant.

7.17.2.3 Test sequences

Table 69 — SBR test sequences

file base name	content	bitrate (kbit/s)	QMF Identification	QMF Accuracy	Envelope Adjuster Accuracy	Grid control tests	Header Change Tests	Inverse Filtering Tests	Additional Sines Tests	CRC	Diff max	RMS max (linear value)	test procedure
al_sbr_twi	none	24	y	y	-	-	-	-	y	-	-	-	-
al_sbr_qmf	Sine Sweep	24	-	y	-	-	-	-	-	-	5	1.4	maxDiff/RMS
al_sbr_e	rectangle * 10Hz sine	24/48	-	-	y	-	-	-	-	y(Note 1)	90	2.0	maxDiff/RMS
al_sbr_gh	rectangle * 10Hz sine	24/48	-	-	-	y	y	-	-	-	51	1.5	maxDiff/RMS
al_sbr_i(Note 2)	rectangle + noise	24/48	-	-	-	-	-	y	-	y(Note 1)	36	3.4	maxDiff/RMS
al_sbr_s	noise	24	-	-	-	-	-	-	y	-	120	1.9	maxDiff/RMS
al_sbr_cm	music	24-128	-	-	-	-	-	-	-	-	-	-	-
al_sbr_sig	music	48	-	-	-	-	-	-	-	-	-	-	-
al_sbr_sr	music	24-56	-	-	-	-	-	-	-	-	-	-	-
al960_sbr_qmf	Sine Sweep	24	-	y	-	-	-	-	-	-	TBD	TBD	maxDiff/RMS
al960_sbr_e	rectangle * 10Hz sine	24/48	-	-	y	-	-	-	-	yA	TBD	TBD	maxDiff/RMS
al960_sbr_gh	rectangle * 10Hz sine	24/48	-	-	-	y	y	-	-	-	TBD	TBD	maxDiff/RMS
al960_sbr_i	rectangle + noise	24/48	-	-	-	-	-	y	-	yA	TBD	TBD	maxDiff/RMS
al960_sbr_s	noise	24	-	-	-	-	-	-	y	-	TBD	TBD	maxDiff/RMS

Note 1: CRC enabled for 32 kHz testvectors

Note 2: The following bitstreams also exist with the suffix _new: al_sbr_i_32_1, al_sbr_i_44_1, al_sbr_i_48_1. These are preferred for conformance testing while the ones without this suffix are deprecated.

7.18 PS (Parametric Stereo)

7.18.1 Compressed data

7.18.1.1 Characteristics

The parametric stereo tool can be used in combination with (the combination of) the AAC LC AOT and the SBR AOT, or with the SSC AOT.

If the parametric stereo tool is used in combination with the AAC LC AOT and the SBR AOT, the PS data shall be stored as outlined in ISO/IEC 14496-3, Annex 8.A, *Combination of the SBR tool with the parametric stereo tool*.

7.18.1.2 Test procedure

Each compressed data shall meet the syntactic and semantic requirements specified in ISO/IEC 14496-3. The decoded data shall also meet the requirements defined in ISO/IEC 14496-3. If a syntactic element is not listed below, no restrictions apply to that element. The **reserved_ps** element shall be encoded with the value zero.

7.18.1.2.1 Compressed MPEG-4 data payload

7.18.1.2.1.1 ps_data()

iid_mode: Shall be encoded with a value in the range of [0 5]. The values 6, 7 are reserved.

icc_mode: Shall be encoded with a value in the range of [0 5]. The values 6, 7 are reserved.

border_position[e]: Shall be encoded with a value in the range of [(border_position[e-1]+1) (*numQMFSlots*-1)] if e>0 or a value in the range of [0 (*numQMFSlots*-1)] if e==0.

iid_dt[]: Shall be encoded with the value 0 if iid_mode of the current envelope e is different from iid_mode of the previous envelope (e-1).

icc_dt[]: Shall be encoded with the value 0 if icc_mode of the current envelope e is different from icc_mode of the previous envelope (e-1).

ps_extension_id: Shall be encoded with the value of 0. The values 1, 2 and 3 are reserved. Only one PS extension of type '00' (IPD/OPD data) may be present in one stereo frame.

7.18.1.2.1.2 ps_extension()

ipd_dt[]: Shall be encoded with the value 0 if iid_mode of the current envelope e is different from iid_mode of the previous envelope (e-1).

opd_dt[]: Shall be encoded with the value 0 if iid_mode of the current envelope e is different from iid_mode of the previous envelope (e-1).

7.18.1.2.1.3 iid_data()

bs_codeword: Shall be encoded with the values listed in the corresponding Huffman table, defined in ISO/IEC 14496-3, Table 8.B.17 or Table 8.B.18.

Conformant compressed MPEG-4 data shall have coded iid_par[e][b] IID indices that are in the range [-7 7] if iid_quant==0 or in the range [-15 15] if iid_quant==1.

7.18.1.2.1.4 icc_data()

bs_codeword: Shall be encoded with the values listed in the corresponding Huffman table, defined in ISO/IEC 14496-3, Table 8.B.19.

Conformant compressed MPEG-4 data shall have coded icc_par[e][b] ICC indices that are in the range [0 7].

7.18.1.2.1.5 ipd_data()

bs_codeword: Shall be encoded with the values listed in the corresponding Huffman table, defined in ISO/IEC 14496-3, Table 8.B.20.

7.18.1.2.1.6 opd_data()

bs_codeword: Shall be encoded with the values listed in the corresponding Huffman table, defined in ISO/IEC 14496-3, Table 8.B.21.

7.18.1.2.1.7 restrictions on underlying coders

The underlying core coder must produce exactly one output channel.

7.18.1.3 HE-AAC v2 profile

if the SBR bitstream element bs_header_flag is 1, enable_ps_header shall be encoded with the value 1 and iid_dt[0], icc_dt[0], ipd_dt[0], opd_dt[0] (if present) shall be encoded with the value 0 and num_env shall not be encoded with the value 0. Only one SBR bitstream element of type EXTENSION_ID_PS may be present in an HE AAC v2 profile stream.

7.18.2 Decoders

7.18.2.1 Characteristics

The object type PS has the Object Type ID 29, and the compressed MPEG-4 data syntax is defined in ISO/IEC 14496-3. The Audio Object Type PS contains the PS Tool.

7.18.2.2 HE-AAC v2 profile

A conformant HE-AAC v2 profile decoder shall support all the abilities of an HE-AAC profile decoder as outlined in 6.6.17.2.1.1 HE-AAC profile.

A conformant HE-AAC v2 profile decoder shall support implicit PS signaling, as outlined in ISO/IEC 14496-3:2009, 1.6.6.3 HE AAC v2 Profile Decoder Behavior in Case of Implicit Signaling.

A conformant HE-AAC v2 profile decoder shall support explicit PS signaling as outlined in ISO/IEC 14496-3, 1.6.6.4 HE AAC v2 Profile Decoder Behavior in Case of Explicit Signaling.

A conformant HE-AAC v2 profile decoder that receives a PS enhanced data stream shall output the mono signal in the two output channels until a first ps_data() element with enable_ps_header==1 is received, ensuring that the PS data can be decoded correctly.

A conformant HE-AAC v2 profile decoder of Level 2 or higher, shall support the baseline version of the PS tool, as outlined in ISO/IEC 14496-3:2009, 8.A.4 *Baseline version of parametric stereo coding tool*.

A conformant HE-AAC v2 profile decoder of Level 3 or higher, shall support mixing mode Ra and Rb, IPD/OPD synthesis and 34 frequency bands resolution, as outlined in ISO/IEC 14496-3:2009, 8.6.4 *Parametric stereo*.

A conformant HE-AAC v2 Profile decoder of a certain level shall always be able to operate the HQ SBR tool for streams containing Parametric Stereo data. For streams not containing Parametric Stereo data, the HE-AAC v2 Profile decoder may operate the HQ SBR tool, or the LP SBR tool.

7.18.2.3 PS conformance test procedure

The conformance test procedure for the PS tool is the same test procedure as for the SBR tool, with the addition that the reference SBR decoder used in the SBR tool test-procedure also includes the PS tool.

7.18.2.4 Test sequences

All bitstreams also exist with the suffix _new. These are preferred for conformance testing while the ones without this suffix are non-conformant and therefore deprecated.

Table 70 — PS test sequences

file base name	content	bitrate (kbit/s)	iid_data	icc_data	ipd_data	opd_data	mixing procedure	iid-/icc-mode	Diff max baseline decoding	RMS max (linear value)	Diff max unrestricted decoding	RMS max (linear value)	test procedure
al_sbr_ps_00	sweep	32	-	-	-	-	-	-	8	1.7	8	1.7	RMS
al_sbr_ps_01	sweep	32	y	-	-	-	-	-	12	1.8	12	1.8	RMS
al_sbr_ps_02	sweep	32	-	y	-	-	Ra	-	30	3.6	30	3.6	RMS
al_sbr_ps_03	sweep	32	y	y	-	-	Rb	-	45	3.7	-	-	RMS
al_sbr_ps_04	sweep	32	y	-	y	y	-	-	12	1.8	-	-	RMS
al_sbr_ps_05	sweep	32	y	y	-	-	Ra	-	40	3.6	-	-	RMS
al_sbr_ps_06	sweep	32	y	y	-	-	Ra	y	40	3.7	-	-	RMS

7.19 SSC (SinuSoidal Coding)

7.19.1 Compressed data

7.19.1.1 Characteristics

Conformant SSC compressed MPEG-4 data shall have the SSC data stored as outlined in ISO/IEC 14496-3 subpart 8 "Technical description of Parametric coding for high quality audio".

7.19.1.2 Test procedure

Each compressed data shall meet the syntactic and semantic requirements specified in ISO/IEC 14496-3. The decoded data shall also meet the requirements defined in ISO/IEC 14496-3. If a syntactic element is not listed below, no restrictions apply to that element. The **reserved** element shall be encoded with the value zero.

7.19.1.2.1 Compressed MPEG-4 data payload

AudioSpecificConfig

audioObjectType: Shall be encoded with the value 28.

samplingFrequencyIndex: Shall be encoded with the value 4.

channelConfiguration: Shall be encoded with a value in the range of [1 2].

SSCSpecificConfig

decoder_level: Shall be encoded with the value 1. The values 0, 2, 3 are reserved.

update_rate: Shall be encoded with the value 4. The values in the ranges [0 3] and [5 7] are reserved.

synthesis_method: Shall be encoded with the value 0. The values 1, 2, 3 are reserved.

mode_ext: Shall be encoded with the value 0 or 1. The values 2, 3 are reserved.

ssc_audio_frame_header

s_nrof_continuations: Shall be encoded with a value not exceeding max_nrof_sinusoids[decoder_level].

s_nrof_den: Shall be encoded with a value not exceeding max_nrof_den[decoder_level].

subframe_transients(sf,ch)

t_loc[sf][ch]: Shall be encoded with a value in the range of [0 S[Update_rate]-1].

t_type[sf][ch]: Shall be encoded with a value in the range of [0 1]. The values 2, 3 are reserved.

t_b_par[sf][ch]: Shall be encoded with a value in the range of [0 3]. The values in the range of [4 7] are reserved.

t_chi_par[sf][ch]: Shall be encoded with a value in the range of [0 3]. The values in the range of [4 7] are reserved.

t_freq[sf][ch][l]: Shall be encoded with a value in the range of [0 485]. The values in the range of [486 511] are reserved.

subframe_sinusoids(sf,ch)

bs_codeword: Shall be encoded with the values listed in the corresponding Huffman tables, defined in ISO/IEC 14496-3, Tables 8.B.1 till 8.B.11, 8.B.15 and 8.B.16.

s_nrof_births[sf][ch]: Shall be encoded with a value not exceeding max_nrof_sinusoids[decoder_level] - s_nrof_confinuations[sf][ch].

subframe_noise(sf,ch)

n_laguerre[ch]: Shall be encoded with a value in the range of [0 2]. The value 3 is reserved.

bs_codeword: Shall be encoded with the values listed in the corresponding Huffman tables, defined in ISO/IEC 14496-3, Tables 8.B.12 till 8.B.14.

ps_data()

iid_mode: Shall be encoded with a value in the range of [0 5]. The values 6, 7 are reserved.

icc_mode: Shall be encoded with a value in the range of [0 5]. The values 6, 7 are reserved.

border_position[e]: Shall be encoded with a value in the range of [(border_position[e-1]+1) (*numQMFSlots*-1)] if e>0 or a value in the range of [0 (*numQMFSlots*-1)] if e==0.

iid_dt[]: Shall be encoded with the value 0 if iid_mode of the current envelope is different from iid_mode of the previous envelope.

icc_dt[]: Shall be encoded with the value 0 if icc_mode of the current envelope is different from icc_mode of the previous envelope.

ps_extension_id: Shall be encoded with the value of 0. The values 1, 2 and 3 are reserved.

ps_extension()

ipd_dt[]: Shall be encoded with the value 0 if iid_mode of the current envelope is different from iid_mode of the previous envelope.

opd_dt[]: Shall be encoded with the value 0 if iid_mode of the current envelope is different from iid_mode of the previous envelope.

reserved_ps: Shall be encoded with the value of 0. The value 1 is reserved.

iid_data()

bs_codeword: Shall be encoded with the values listed in the corresponding Huffman table, defined in ISO/IEC 14496-3, Table 8.B.17 or Table 8.B.18.

Conformant compressed MPEG-4 data shall have coded iid_par[e][b] IID indices that are in the range [-7 7] if iid_quant==0 or in the range [-15 15] if iid_quant==1.

icc_data()

bs_codeword: Shall be encoded with the values listed in the corresponding Huffman table, defined in ISO/IEC 14496-3, Table 8.B.19.

Conformant compressed MPEG-4 data shall have coded icc_par[e][b] ICC indices that are in the range [0 7].

ipd_data()

bs_codeword: Shall be encoded with the values listed in the corresponding Huffman table, defined in ISO/IEC 14496-3, Table 8.B.20.

opd_data()

bs_codeword: Shall be encoded with the values listed in the corresponding Huffman table, defined in ISO/IEC 14496-3, Table 8.B.21.

7.19.2 Decoders

7.19.2.1 Characteristics

The object type SSC has the Object Type ID 28, and the compressed MPEG-4 data syntax is defined in ISO/IEC 14496-3. The Audio Object Type SSC contains the SSC and PS tools.

7.19.2.2 SSC levels

A conformant decoder supporting the SSC Object Type of Level 2 or higher, shall support the baseline version of the PS tool, as outlined in ISO/IEC 14496-3.

A conformant decoder supporting the SSC Object Type of Level 3 or higher, shall support IPD/OPD synthesis and 34 frequency bands resolution, as outlined in ISO/IEC 14496-3.

A conformant decoder supporting the SSC Object Type of a certain level shall always be able to operate the SSC tool for streams containing Parametric Stereo data. For streams not containing Parametric Stereo data, the decoder supporting the SSC Object Type shall operate the SSC tool.

The specifications given in the level definitions must be met. The output signal of the decoder under test must meet the criteria for accuracy of deterministic signal components and statistical properties of stochastic signal components as described below. Two alternative accuracy classes for SSC decoders are defined:

Full Accuracy SSC decoder

Fixed Point Accuracy SSC decoder

The criteria for deterministic signal components depend on the accuracy class selected. The criteria for stochastic signal components are the same for both accuracy classes.

7.19.2.3 SSC conformance test procedure

Test compressed data and reference decoder output signals are provided to apply the different conformance criteria using the procedures described in the following Subclauses. Software implementing the different test procedures will be made available.

Since Level 2 includes Level 1, a Level 2 conforming decoder must also meet the criteria for Level 1. A Level 3 conforming decoder must also meet the criteria for Level 1.

The conformance of the SSC decoder tools can be checked with compressed data for the SSC object type.

To be called a conforming SSC decoder, the required conformance criteria must be met for all test compressed data listed in 7.19.2.4, that are applicable at the selected Level. The conformance criteria for deterministic signal components depend on the selected accuracy class.

7.19.2.3.1 Conformance criterion for deterministic components of Full Accuracy SSC decoders

A Full Accuracy SSC decoder at an accuracy level of "K bit" has to fulfil the RMS/LSB criterion as defined in 7.1.2.2.1.

7.19.2.3.2 Conformance criteria for deterministic components of Fixed-Point Accuracy SSC decoders

The conformance criteria for Fixed-Point Accuracy decoders are based on measuring the segmental SNR and the LPC Cepstral Distortion (CD) between the reference decoder output and the output of the decoder to be tested. The segment length to be used in the calculation of the SNR is equal to the audio frame length (i.e. 8*384=3072 samples). The SNR and the CD have to be calculated only for those segments for which the power of the reference waveform is in the range [-50...-5] dB. CD is defined as

$$\mathrm{CD} = \frac{10}{\ln(10)} \cdot \sqrt{2\mathrm{D}} \ .$$

D is the accumulated distortion of the LPC cepstrum C_{ref} of the reference waveform and C_{test} of the output of the decoder under test. D is defined as

$$\mathrm{D} = \sum_{i=1}^{\mathrm{N}} \left(C_{\mathrm{ref}}[i] - C_{\mathrm{test}}[i]\right)^2 \ .$$

N is the LPC cepstrum order which equals 32. The LPC cepstrum C[i] is defined by means of the algorithm `lpc2cepstrum` based on the LPC coefficients of a 16th order linear prediction filter. The computation of the LPC filter coefficients lpc_coef [j] is defined by the algorithm `calculate_lpc` (as defined in ISO/IEC 14496-3).

For a Fixed-Point Accuracy SSC decoder, the average value of the segmental SNR shall exceed 30 dB. At the same time, the average value of the CD shall not exceed 1 dB.

7.19.2.3.3 Conformance criteria for noise generators and spectral noise envelopes of SSC decoders

The applied noise generator must not show a periodicity of less than one second. The average spectral envelope of a stationary noise component must meet a Cepstral Distance criterion when compared to the reference spectral envelope.

To perform this test, the noise signal is re-whitened by a prediction filter, which is inverse to the filter used in the noise synthesis of the decoder. The required filter parameters are given in the parameter file accompanying a test compressed data. The autocorrelation function (ACF) of the re-whitened noise is calculated over a sufficiently long noise signal (e.g. 20 seconds, i.e. 288 frames) and normalized by dividing all values by the zero-th value of the ACF. The absolute values of this normalized ACF must not exceed a limit of e.g. 0.1 in the range of e.g. 1...15999.

In addition, the average power of the analyzed segment must be in the range of e.g. +/- 0.5 dB relative to the given reference power.

Test stream ssc_n1 must be used to perform the noise generator and spectral noise envelope test. This test stream only contains noise tool related information. The HILN conformance tool *hilnconf* shall be applied in the following way:

hilnconf ssc_n1.ctp test.wav 2

Where “test.wav” is the decoded output of the decoder under test for test stream ssc_n1.

7.19.2.3.4 Conformance criteria for temporal noise envelopes of SSC decoders

The average of multiple instances of the same temporal envelope must closely enough resemble the reference temporal envelope.

To perform this test, the signal is cut into segments with a length of 2 frames. For every sample position in this set of segments, the squares of the sample values of all segments are accumulated and afterwards divided by the number of segments to calculate the average power for each sample position. The square roots of the resulting values must not differ from the reference temporal amplitude envelope by more than e.g. +/-20% of the nominal noise amplitude.

Test stream ssc_n2 must be used to perform the temporal noise envelope test. This test stream only contains noise tool related information.

7.19.2.3.5 Conformance criteria for tempo and pitch scaling of SSC decoders

Test streams ssc_s2, ssc_n1, ssc_n2 and ssc_ps6 may be used to perform the tempo and pitch scaling test. For these four test streams reference decoded output signals are provided for a subframe size of 360 samples and a pitch scaling factor of 0.9. For test stream ssc_s2 the deterministic components tests apply. For the test streams ssc_n1 and ssc_n2, the spectral and temporal noise envelope tests apply, respectively. For test stream ssc_n1 a corresponding ssc_n1_scaling_360_0.9.ctp test parameter file is provided and can be used as described in Section 7.11.2.2.3.

7.19.2.3.6 Conformance criteria for Parametric Stereo decoding of SSC decoders

For the test streams containing Parametric Stereo data also accompanying test streams containing only the mono parameters are provided. Furthermore a modified version of the reference decoder is prepared that is able to replace the SSC mono signal by a signal read from a file before processing the Parametric Stereo data.

To perform this test, the decoder output for the accompanying test stream is provided as the external input to the modified reference decoder. The output of the modified reference decoder is compared with the output of the decoder under test according to the conformance criteria for deterministic components.

This is illustrated in Figure 12.

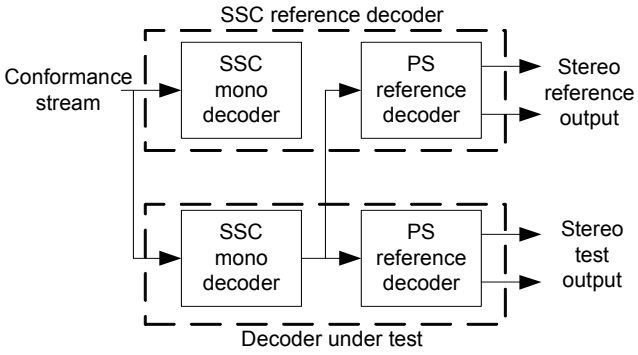

Figure 12 — Parametric stereo conformance setup.

7.19.2.4 Proposed test sequences

Table 71 — SSC test sequences

file base name	content	bitrate (kbit/s)	channel Configuration	mode_ext	Refresh frames test	phase_jitter	Transient type test	Frequency granularity test	Amplitude granularity test	Laguerre filtering	n_overlap_lsf	iid_mode	icc_mode	ipdopd	PS configuration switching	Transients	Sinusoids	Noise	Parametric stereo coding	Conformance criteria
ssc_t1	test signal	12	1	-	-	-	0	-	-	-	-	-	-	-	-	y	y	-	-	Deterministic
ssc_t2	test signal	6	1	-	-	-	1	-	-	-	-	-	-	-	-	y	-	-	-	Deterministic
ssc_s1	test signal	18	1	-	-	-	-	-	-	-	-	-	-	-	-	-	y	-	-	Deterministic
ssc_s2	test signal	3	1	-	y	y	-	y	y	-	-	-	-	-	-	-	y	-	-	Deterministic
ssc_s3	test signal	30	2	0	-	-	-	-	-	-	-	-	-	-	-	-	y	-	-	Deterministic
ssc_n1	test signal	4	1	-	-	-	-	-	-	y	y	-	-	-	-	-	-	y	-	Spectral noise envelope
ssc_n2	test signal	5	1	-	-	-	-	-	-	-	y	-	-	-	-	-	-	y	-	Temporal noise envelope
ssc_n3	test signal	5	1	-	y	-	-	-	-	y	y	-	-	-	-	-	-	y	-	subjective
ssc_n4	test signal	42	2	0	-	-	-	-	-	-	-	-	-	-	-	y	y	y	-	subjective
ssc_ps1	music	24	2	1	-	-	-	-	-	-	-	0	-	-	-	y	y	y	y	Deterministic
ssc_ps2	music	24	2	1	-	-	-	-	-	-	-	5	-	-	-	y	y	y	y	Deterministic
ssc_ps3	music	24	2	1	-	-	-	-	-	-	-	-	1	-	-	y	y	y	y	Deterministic
ssc_ps4	music	24	2	1	-	-	-	-	-	-	-	-	5	-	-	y	y	y	y	Deterministic
ssc_ps5	music	24	2	1	-	-	-	-	-	-	-	4	4	-	-	y	y	y	y	Deterministic
ssc_ps6	music	24	2	1	-	-	-	-	-	-	-	2	2	y	-	y	y	y	y	Deterministic
ssc_ps7	music	24	2	1	-	-	-	-	-	-	-	y	y	y	y	y	y	y	y	Deterministic

7.20 DST (Lossless coding of oversampled audio)

7.20.1 Compressed data

7.20.1.1 Characteristics

Conformant DST compressed MPEG-4 data shall have the DST data stored as outlined in ISO/IEC 14496-3.

7.20.1.2 Test procedure

Each compressed data shall meet the syntactic and semantic requirements specified in ISO/IEC 14496-3. The decoded data shall also meet the requirements defined in ISO/IEC 14496-3. If a syntactic element is not listed below, no restrictions apply to that element. The **reserved** element shall be encoded with the value zero.

7.20.1.2.1 Compressed MPEG-4 data payload

7.20.1.2.1.1 AudioSpecificConfig

audioObjectType: Shall be encoded with the value 35.

samplingFrequencyIndex: Shall be encoded with the value 0xf.

SamplingFrequency: Shall be encoded with the value 64*44100 or 128*44100 or 256*44100.

channelConfiguration: Shall be encoded with the value 0.

7.20.1.2.1.2 DSTSpecificConfig

N_Channels: Shall not be encoded with the value 0.

7.20.1.2.1.3 DST

DST_X_Bit: Shall be encoded with the value '0'.

7.20.1.2.1.4 Channel_Segmentation

Nr_Of_Segments: Shall not exceed the value of 4 for the Filter_Segmentation and the Filter_And_Ptable_Segmentation, and shall not exceed the value of 8 for the Ptable_Segmentation.

Resolution: Shall be encoded with a value in the range of [1 Frame_length-MINSEGLEN], with MINSEGLEN as defined in ISO/IEC 14496-3:2009, 10.6.2.1.2.5.2.2.2.
Scaled_Length[*Nr_Of_Segments*]: Shall be encoded with a value in the range of [1 Range], with Range as defined in ISO/IEC 14496-3:2009, 10.6.2.1.2.5.2.2.3.

7.20.1.2.1.5 Filter_Coef_Sets

CC_Method: Shall not be encoded with the value '11'.

CCM: Shall not be encoded with the value 7.

7.20.1.2.1.6 Probability_Tables

PC_Method: Shall not be encoded with the value '11'.

PCM: Shall be encoded with a value in the range of [0 4].

7.20.2 Decoders

7.20.2.1 Characteristics

The object type DST has the Object Type ID 35, and the compressed MPEG-4 data syntax is defined in ISO/IEC 14496-3. The Audio Object Type DST contains the DST tool.

7.20.2.2 DST conformance test procedure

Test compressed data and reference decoder output signals are provided to apply the conformance criterion using the procedure described in the following Subclause.

The conformance of the DST decoder tool can be checked with compressed data for the DST object type.

For lossless compressed data the conformance criterion is bit exact reproduction of the reference decoder output, this means that all bits in the output of the test decoder are identical to the corresponding bits in the output of the reference decoder.

To be called a conforming DST decoder, the required conformance criterion must be met for all test compressed data listed in 7.20.2.3.

7.20.2.3 Proposed test sequences

Table 72 — DST test sequences

file base name	content	Number of Channels	64 * 44100	128 * 44100	256 * 44100	Plain DSD frames	Segmentation Test	Mapping Test	Filter_Coef_Sets Test	Probability_Tables Test	Arithmetic_Coded_Data Test	Conformance criteria
dst_00	test signal	1, 2, 3, 4, 5, 6, 8, 16, 32, 64	y	y	y	-	-	-	-	-	-	Bit exact
dst_01	test signal	2, 5, 6	y	y	y	y	-	-	-	-	-	Bit exact
dst_02	test signal	2, 5, 6	y	y	y	-	y	-	-	-	-	Bit exact
dst_03	test signal	2, 5, 6	y	-	-		-	y	-	-	-	Bit exact
dst_04	test signal	2, 5, 6	y	-	-		-	-	y	-	-	Bit exact
dst_05	test signal	2, 5, 6	y	-	-		-	-	-	y	-	Bit exact
dst_06	test signal	2, 5, 6	y	-	-		-	-	-	-	y	Bit exact

7.20.3 Encoders

In order to guarantee that for an encoder implementation that the decoded output results in an exact replica of the input signal, the following procedure should be followed:

- Generate bitstreams using the target encoder for the conformance test item reference data

- Decode these bitstreams using the conformant reference SW decoder;

- Verify that the decoded outputs are identical to the inputs.

7.21 Layer-3

7.21.1 Compressed data

7.21.1.1 Characteristics

Conformant Layer-3 compressed MPEG-4 data shall have the Layer-3 data stored as specified in ISO/IEC 14496-3, subpart 9.

7.21.1.2 Test procedure

Each compressed data shall meet the syntactic and semantic requirements specified in ISO/IEC 14496-3. The decoded data shall also meet the requirements defined in ISO/IEC 11172-3 and ISO/IEC 13818-3. If a syntactic element is not listed below, no restrictions apply to that element. The **reserved** element shall be encoded with the value zero.

7.21.1.2.1 AudioSpecificConfig

audioObjectType: Shall be encoded with the value 34.

7.21.1.2.2 Bitstream payload

syncword: shall be set to the total length of the mp3_channel_element (consisting of header, error_check, side_info and main data) in bytes

main_data_begin (9/8 bit): shall be set to the correct value for the corresponding MPEG-1/2 Layer 3 bitstream, or to zero

main_data(): shall be stored immediately following the side information

7.21.2 Decoders

7.21.2.1 Characteristics

The MPEG-4 Layer-3 object type (“mp3on4”) is the counterpart to the MPEG-1/2 Layer-3, though also offering multi-channel support and support for lower sampling rates. The compressed MPEG-4 data syntax is defined in ISO/IEC 14496-3. The Audio Object Type Layer-3 contains re-formatted ISO/IEC 11172-3 or ISO/IEC 13818-3 Layer 3 compressed data.

7.21.2.2 Test procedure

The test procedures specified in ISO/IEC 11172-4 and ISO/IEC 13818-4 shall be applied. If no test is specified, a check of conformance using appropriate measurements, e.g. the LSB criterion or objective perceptual measurement systems, is not mandatory but highly recommended.

7.21.2.3 Test sequences

To test MPEG-4 Layer-3 decoders, ISO/IEC JTC 1/SC 29/WG 11 supplies a number of test sequences. The test sequences are defined in Table 73.

Table 73 — MPEG-1/2 Audio on MPEG-4 Layer-3 test sequences

file base name	mnemonic	origin	content	IDex	ID	bitrate_index	sampling_rate	mode	JS	CRC	test procedure
l3_test10101	he_32khz	11172-4	sine	1	1	0001	10	11	none	x	none
l3_test10102	he_44khz	11172-4	sine	1	1	0001	00	11	none	x	none
l3_test10103	he_48khz	11172-4	sine	1	1	0001	01	11	none	x	none
l3_test10104	si	11172-4	noise	1	1	0101	00	11	none	x	none
l3_test10105	si_block	11172-4	noise	1	1	0101	00	11	none	x	none
l3_test10106	si_huff_patched	derived from si_huff (11172-4)	noise	1	1	0101	00	11	none	x	none
l3_test10107	mp3on4_1	new	music	1	1	0101	01	11	none	none	none
l3_test10201	he_mode	11172-4	sine	1	1	1001	00	all	MS, IS	x	none
l3_test10202	hecommon	11172-4	sine	1	1	1001	00	00	none		none
l3_test10203	intens	new	noise	1	1	1001	00	01	IS	x	none
l3_test10204	sin1k0db	11172-4	sine	1	1	1001	00	01	none	x	none
l3_test10205	worstcas	new	artificial	1	1	1110	00	01	none	x	none
l3_test10206	compl	11172-4	sweep (10 Hz …10 kHz, -20 dB)	1	1	0101	01	11	none	x	RMS
l3_test10207	mp3on4_2	new	music	1	1	1001	01	01	MS	none	none
l3_test10501	mp3on4_5	new	music	1	1	1110	01	11,01,01	MS	none	none
l3_test20140	bitrate_16_all	13818-4	tonal, sine	1	0	0001	10	11	none	x	none
l3_test20141	bitrate_22_all	13818-4	tonal, sine	1	0	0001	00	11	none	x	none
l3_test20142	bitrate_24_all	13818-4	tonal, sine	1	0	0001	01	11	none	x	none
l3_test20143	compl24	13818-4	sweep (10 Hz …10 kHz, -20 dB)	1	0	1100	01	11	none	x	RMS
l3_test20244	noise	13818-4	white noise	1	0	1010	00	01	none	x	none
l3_test20245	is2	13818-4	white noise	1	0	1110	00	01	IS	x	none
l3_test20246	ms2	13818-4	white noise	1	0	1110	00	01	MS, IS	x	none
l3_test20247	Test_20_00	new	sweep	1	0	1010	00	01	IS	x	none
l3_test20248	Test_20_80	new	sweep	1	0	1010	00	01	IS	x	none
l3_test25101	compl12	new	sweep (10 Hz …5 kHz, -20 dB)	0	0	1000	01	11	none	x	RMS
l3_test25102	bitrate_8_all	new	sine	0	0	0000	10	11	none	x	none
l3_test25103	bitrate_11_all	new	sine	0	0	0000	00	11	none	x	none
l3_test25104	bitrate_12_all	new	sine	0	0	0000	01	11	none	x	none
l3_test25201	noise	new	noise	0	0	1000	00	01	none	x	none
l3_test25202	test20_00	new	sweep (10 Hz …5 kHz)	0	0	1000	00	01	none	x	none

file base name	mnemonic	origin	content	IDex	ID	bitrate_index	sampling_rate	mode	JS	CRC	test procedure
l3_test25203	test20_80	new	sweep (10 Hz …5 kHz)	0	0	1000	00	01	none	x	none
l3_test25204	bitrates_8i	new	noise	0	0	1000	10	01	IS	x	none
l3_test25205	bitrates_11i	new	noise	0	0	1000	00	01	MS, IS	x	none
l3_test25206	bitrates_12i	new	noise	0	0	1000	01	01	MS, IS	x	none
l3_test25207	sw_8i	new	sweep (550 Hz … 5 kHz)	0	0	0100	10	01	IS	x	none
l3_test25208	sw_11i	new	sweep (550 Hz … 5 kHz)	0	0	0100	00	01	IS	x	none
l3_test25209	sw_12i	new	sweep (550 Hz … 5 kHz)	0	0	0100	01	01	IS	x	none

7.22 ALS (Audio lossless coding)

7.22.1 Compressed data

7.22.1.1 Characteristics

Conformant ALS compressed MPEG-4 data shall have the ALS data stored as outlined in ISO/IEC 14496-3.

7.22.1.2 Test procedure

Each compressed data shall meet the syntactic and semantic requirements specified in ISO/IEC 14496-3. The decoded data shall also meet the requirements defined in ISO/IEC 14496-3. If a syntactic element is not listed below, no restrictions apply to that element. The **reserved** element shall be encoded with the value zero.

7.22.1.2.1 Compressed MPEG-4 data payload

7.22.1.2.1.1 AudioSpecificConfig

audioObjectType: Shall be encoded with the value 36.

samplingFrequencyIndex: Shall be encoded with the value 0xf.

SamplingFrequency: Shall be encoded with the value samp_freq if samp_freq < 2^{24}, and with the value 0 otherwise, with samp_freq as defined in ISO/IEC 14496-3.

channelConfiguration: Shall be encoded with the value 0.

7.22.1.2.1.2 ALSSpecificConfig

als_id: Shall be encoded with the value 1095521024 (0x414C5300 hex).

file_type: Shall be encoded with a value in the range [0 3]. Other values are reserved.

resolution: Shall be encoded with a value in the range [0 3]. Other values are reserved.

frame_length: Shall be encoded with a value that is divisible by 2^{levels} without remainder, with levels as defined in ISO/IEC 14496-3.

ra_flag: Shall be encoded with a value in the range [0 2]. The value 3 is reserved.

7.22.1.2.1.3 block_data

opt_order: Shall be encoded with a value in the range [0 Kmax], with Kmax as defined in ISO/IEC 14496-3.

7.22.2 Decoders

7.22.2.1 Characteristics

The object type ALS has the Object Type ID 36, and the compressed MPEG-4 data syntax is defined in ISO/IEC 14496-3. The Audio Object Type ALS contains the ALS tools.

7.22.2.2 ALS conformance test procedure

Compressed data and reference decoder output signals are provided to apply the conformance criterion using the procedure described in the following Subclause.

The conformance of the ALS decoder can be checked with compressed data for the ALS object type.

For lossless compressed data the conformance criterion is bit exact reproduction of the reference decoder output, this means that all bits in the output of the test decoder are identical to the corresponding bits in the output of the reference decoder.

To be called a conforming ALS decoder, the required conformance criterion must be met for all compressed data listed in 7.22.2.3.

7.22.2.3 Proposed test sequences

Table 74 — ALS test sequences

file base name	content	Number of Channels	sampling frequency (kHz)	word length (bit)	adaptive order	random access	block switching	LTP	joint stereo	MCC	BGMC	RLSLMS	Conformance criteria
als_00	music	2	48, 96,192	16, 20, 24	y	y							Bit exact
als_01	music	2	48, 96,192	16, 20, 24	y		y		y				Bit exact
als_02	music	2	48, 96,192	16, 20, 24				y					Bit exact
als_03	music	2	48, 96,192	16, 20, 24					y				Bit exact
als_04	music	2	48, 96,192	16, 20, 24						y			Bit exact
als_05	music	2	48, 96,192	16, 20, 24							y		Bit exact
als_06	music	2	48, 96,192	16, 20, 24								y	Bit exact
als_07	music	2	192	32 float	y								Bit exact
als_08	music	6	96	24	y	y	y	y	y	y	y		Bit exact
als_09	bio data	512	2	16	y		y	y	y	y	y		Bit exact
als_10	sine wave	1	48	16	y	y							Bit exact

Note: Test sequence “als_10” is only necessary to check for conformant decoding of the OAFI box (see 7.22.4).

7.22.3 Encoders

In order to guarantee for an encoder implementation that the decoded output results in an exact replica of the input signal, the following procedure should be followed:

- Generate bitstreams using the target encoder for the conformance test item reference waveform data;

- Decode these bitstreams using the conformant reference software decoder;

- Verify that the decoded outputs are identical to the inputs.

7.22.4 OAFI box

The Original Audio File Information Box permits the storage of the ancillary (non-audio) data of an original audio file in an ISO Base Media File format file. This file would typically also contain the compressed audio data of the original audio file. This box is particularly useful in combination with lossless audio coding when restoration of the original input audio file is of interest.

Therefore, the OAFI conformance can be checked by means of compressed data for the ALS object type that additionally contains an OAFI box. Such data is defined by test sequence "als_10" in 7.22.2.3 above, Table *74*.

7.23 SLS (Scalable Lossless Coding)

7.23.1 Compressed data

7.23.1.1 Characteristics

Conformant SLS compressed MPEG-4 data shall have the SLS data stored as outlined in ISO/IEC 14496-3.

7.23.1.2 Test procedure

Each compressed data shall meet the syntactic and semantic requirements specified in ISO/IEC 14496-3. The decoded data shall also meet the requirements defined in ISO/IEC 14496-3. If a syntactic element is not listed below, no restrictions apply to that element. The **reserved** element shall be encoded with the value zero.

7.23.1.2.1 Compressed MPEG-4 data payload

7.23.1.2.1.1 AudioSpecificConfig

audioObjectType: Shall be encoded with the value 37 (core mode) or 38 (non-core mode).

7.23.1.2.1.2 SLSSpecificConfig

pcmWordLength: Shall be encoded with a value in the range [0 3], as defined in ISO/IEC 14496-3. Other values are reserved.

aac_core_present: either 0 or 1 for AOT 37, 0 for AOT 38.

frameLength: Shall be encoded with a value in the range [0 3], as defined in ISO/IEC 14496-3. Other values are reserved.

7.23.1.2.1.3 lle_header()

band_type_signaling: Shall be encoded with a value in the range [0 2], as defined in ISO/IEC 14496-3. The value of 3 is reserved.

7.23.2 Decoders

7.23.2.1 Characteristics

The object type SLS has either the Object Type ID 37 (core mode) or 38 (non-core mode), and the compressed MPEG-4 data syntax is defined in ISO/IEC 14496-3. The Audio Object Type SLS contains the SLS tools.

7.23.2.2 Test procedure

Compressed data and reference decoder output signals are provided to apply the conformance criterion using the procedure described in the following Subclause. Note that for the bitstreams providing lossless reconstruction the original input signals provide the reference for the decoder output signal.

The conformance of the SLS decoder can be checked with compressed data for the SLS object type.

For lossless compressed data the conformance criterion is bit exact reproduction of the reference decoder output, this means that all bits in the output of the test decoder are identical to the corresponding bits in the output of the reference decoder.

To be called a conforming AAC-LC+SLS decoder, the required conformance criterion must be met for all compressed data with core coder AOT2 listed in 7.23.2.3.

To be called a conforming BSAC+SLS decoder, the required conformance criterion must be met for all compressed data with core coder AOT22 listed in 7.23.2.3.

To be called a conforming non-core SLS decoder, the required conformance criterion must be met for all compressed data with no core coder listed in 7.23.2.3.

7.23.2.3 Test sequences

Table 75 — SLS test sequences

file base name	content	number of channels	sampling frequency (kHz)/ word length (bit)	max. truncation bitrate (kbit/s/ch)	core coder (kbit/s/ch)	BPGC	CBAC	conformance criterion
sls2100	mixed	2	48 / 16, 48 / 24 96 / 24, 192 / 24	none (lossless)	AOT 2 (64)	X		bit exact
sls2101	mixed	2	48 / 16, 48 / 24 96 / 24, 192 / 24	none (lossless)	AOT 2 (64)		X	bit exact
sls2110	mixed	2	48 / 16, 48 / 24 96 / 24, 192 / 24	256	AOT 2 (64)	X		bit exact
sls2111	mixed	2	48 / 16, 48 / 24 96 / 24, 192 / 24	256	AOT 2 (64)		X	bit exact
sls2200	mixed	2	48 / 16, 48 / 24 96 / 24, 192 / 24	none (lossless)	AOT 22 (64)	X		bit exact

file base name	content	number of channels	sampling frequency (kHz)/ word length (bit)	max. truncation bitrate (kbit/s/ch)	core coder (kbit/s/ch)	BPGC	CBACC	conformance criterion
sls2201	mixed	2	48 / 16, 48 / 24 96 / 24, 192 / 24	none (lossless)	AOT 22 (64)		X	bit exact
sls2210	mixed	2	48 / 16, 48 / 24 96 / 24, 192 / 24	256	AOT 22 (64)	X		bit exact
sls2211	mixed	2	48 / 16, 48 / 24 96 / 24, 192 / 24	256	AOT 22 (64)		X	bit exact
sls2300	mixed	2	48 / 16, 48 / 24 96 / 24, 192 / 24	none (lossless)	no core	X		bit exact
sls2301	mixed	2	48 / 16, 48 / 24 96 / 24, 192 / 24	none (lossless)	no core		X	bit exact
sls2310	mixed	2	48 / 16, 48 / 24 96 / 24, 192 / 24	256	no core	X		bit exact
sls2311	mixed	2	48 / 16, 48 / 24 96 / 24, 192 / 24	256	no core		X	bit exact
sls6100	mixed	6	48 / 24 96 / 24,	none (lossless)	AOT 2 (64)	X		bit exact
sls6101	mixed	6	48 / 24 96 / 24,	none (lossless)	AOT 2 (64)		X	bit exact
sls6110	mixed	6	48 / 24 96 / 24,	256	AOT 2 (64)	X		bit exact
sls6111	mixed	6	48 / 24 96 / 24,	256	AOT 2 (64)		X	bit exact
sls6200	mixed	6	48 / 24 96 / 24,	none (lossless)	AOT 22 (64)	X		bit exact
sls6201	mixed	6	48 / 24 96 / 24,	none (lossless)	AOT 22 (64)		X	bit exact
sls6210	mixed	6	48 / 24 96 / 24,	256	AOT 22 (64)	X		bit exact
sls6211	Mixed	6	48 / 24 96 / 24,	256	AOT 22 (64)		X	bit exact
sls6300	mixed	6	48 / 24 96 / 24,	none (lossless)	no core	X		bit exact
sls6301	mixed	6	48 / 24 96 / 24,	none (lossless)	no core		X	bit exact
sls_6310	Mixed	6	48 / 24 96 / 24,	256	no core	X		bit exact
sls_6311	mixed	6	48 / 24 96 / 24,	256	no core		X	bit exact

7.23.3 Encoders

In order to guarantee for an encoder implementation that the decoded output results in an exact replica of the input signal, the following procedure should be followed:

- Generate bitstreams using the target encoder for the conformance test item reference waveform data

- Decode these bitstreams using the conformant reference software decoder;

- Verify that the decoded outputs are identical to the inputs.

7.24 Layer-1 and Layer 2

7.24.1 Compressed data

7.24.1.1 Characteristics

Conformant Layer-1 and Layer 2 compressed MPEG-4 data shall have the Layer-1 and Layer 2 data stored as specified in ISO/IEC 14496-3, subpart 9.

7.24.1.2 Test procedure

Each compressed data shall meet the syntactic and semantic requirements specified in ISO/IEC 14496-3. The decoded data shall also meet the requirements defined in ISO/IEC 11172-3 and ISO/IEC 13818-3.If a syntactic element is not listed below, no restrictions apply to that element. The **reserved** element shall be encoded with the value zero.

7.24.1.2.1 AudioSpecificConfig

audioObjectType: Shall be encoded with the value 32 for Layer 1 and the value 33 for Layer 2.

7.24.2 Decoders

7.24.2.1 Characteristics

The MPEG-4 Layer-1 and Layer 2 object types are the counterpart to the MPEG-1/2 Layer-1 and Layer 2 respectively. The compressed MPEG-4 data syntax is defined in ISO/IEC/14496-3. The Audio Object Types Layer-1 and Layer 2 contains ISO/IEC 11172-3 or ISO/IEC 13818-3 Layer 1 and Layer 2 compressed data.

7.24.2.2 Test procedure

The test procedures specified in 11172-4 and 13818-4 shall be applied. If no test is specified, a check of conformance using appropriate measurements, e.g. the LSB criterion or objective perceptual measurement systems, is not mandatory but highly recommended.

7.24.2.3 Test sequences

To test MPEG-4 Layer-1 and Layer 2 decoders, ISO/IEC JTC 1/SC 29/WG 11 supplies a number of test sequences. The test sequences are defined in Table 76 —MPEG-1/2 Audio on MPEG-4 Layer-1 and Layer 2 test sequences.

Table 76 —MPEG-1/2 Audio on MPEG-4 Layer-1 and Layer 2 test sequences

file base name	Layer	Origin	content	bitrate_index	sampling_rate	mode	CRC	test procedure
l1_fl1	I	11172-4	sweep	1100	10	01	yes	none
l1_fl2	I	11172-4	sweep	1100	00	01	yes	none
l1_fl3	I	11172-4	sweep	1100	01	01	yes	none
l1_fl4	I	11172-4	sweep	0001	10	11	no	none
l1_fl5	I	11172-4	sine	1110	01	10	yes	none
l1_fl6	I	11172-4	noise	1100	00	00	yes	none
l1_fl7	I	11172-4	special	1100	00	00	yes	none
l1_fl8	I	11172-4	sweep	1100	00	00	no	RMS
l2_fl10	II	11172-4	sweep	1010	10	00,01	yes	none
l2_fl11	II	11172-4	sweep	1010	00	00,01	yes	none
l2_fl12	II	11172-4	sweep	1010	01	00,01	yes	none
l2_fl13	II	11172-4	sweep	0001	10	11	no	none
l2_fl14	II	11172-4	sine	1110	01	10	yes	none
l2_fl15	II	11172-4	noise	1110	01	00	yes	none
l2_fl16	II	11172-4	sine	1100	01	00	yes	none
l2_test24	II	13818-4	sweep	1010	10	01	yes	none
l2_test25	II	13818-4	seep	1010	00	01	yes	none
l2_test26	II	13818-4	sweep	1010	01	01	yes	none
l2_test27	II	13818-4	sweep	0010	01	01	yes	none
l2_test28	II	13818-4	sweep	1010	01	01	yes	none
l2_test29	II	13818-4	sweep	1110	01	01	yes	none
l2_test30	II	13818-4	sweep	1010	01	11	yes	none
l2_test31	II	13818-4	sweep	1010	01	10	yes	none
l2_test32	II	13818-4	sweep	1010	01	00	no	none
l1_test33	I	13818-4	sweep	1100	00	01	yes	none
l2_test34	II	13818-4	sweep	1110	01	00	yes	RMS

7.25 Low Delay SBR

7.25.1 Compressed Data

7.25.1.1 Compressed MPEG-4 data payload

The same data payload for compressed data as outlined in 7.17.1.2.1 for SBR applies for LD-SBR, apart from the subsequent syntax elements:

7.25.1.1.1 Syntax of low_delay_sbr_data()

bs_sbr_crc_bits: no restrictions apply

bs_header_flag: no restrictions apply

7.25.1.1.2 sbr_ld_grid()

bs_frame_class: no restrictions apply

tmp: no restrictions apply

bs_amp_res: no restrictions apply

bs_freq_res: no restrictions apply

bs_transient_position: no restrictions apply for a framelength of 512; for 480 shall not exceed 15.

7.25.2 Decoders

7.25.2.1 Characteristics

The same characteristics of the decoder as outlined in 7.17.2.1 for SBR apply for LD-SBR.

7.25.2.1.1 LD-SBR conformance test procedure

The conformance test procedure internally creates a reference for comparison, given an input compressed MPEG-4 data file and the output from the decoder under test.

The conformance test procedure stipulate:

- reading the compressed MPEG-4 data file;
- decode the data file and store it as the reference to be compared against;
- read the wave output from the decoder under test and compare it with the stored reference using maximum amplitude difference and RMS criteria (see Table 77 — ELD with LD-SBR test sequences).

If a complex QMF filter bank with a modified internal phase angle (hereinafter referred to as twiddles) is used in the decoder under test, the er_eld_sbr_twi_* sequences shall be tested first. Contrary to the other er_eld_sbr_* test-sequences, these sequences consist of zero signal AAC ELD core data plus SBR elements that trigger sine-addition in the entire frequency range covered by LD-SBR. The output-signal of this test sequence will contain sinusoids with phase angles that depend on the implementation of the synthesis filterbank in the decoder under test. Based on the obtained output signal from the decoder under test, the two parameters φ and β describing the phase characteristics of the filter banks in the decoder under test are identified. Since different phase characteristics may be used, depending on whether downsampled LD-SBR is used or not, both the er_eld_sbr_twi_24* and the er_eld_sbr_twi_48* sequence must be run prior to conformance testing. The file sequence is only used to perform an examination and identification of the filter bank. The twiddle estimation is identical to that described in 7.17 SBR with the exception that k0 is incremented by two and k2 is decremented by two. Figure 13 — Block diagram of the LD-SBR conformance test procedure outlines the LD-SBR conformance test procedure.

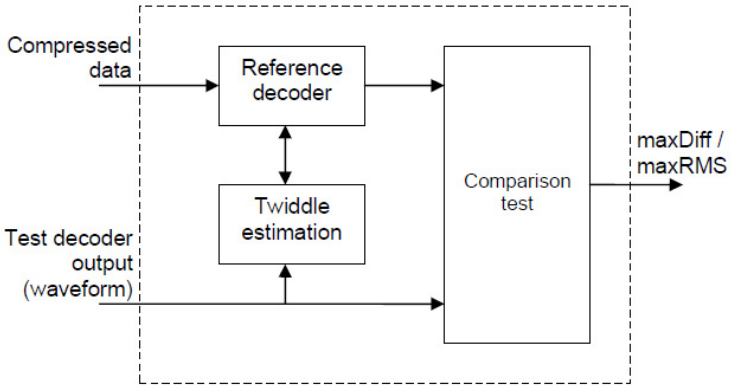

Figure 13 — Block diagram of the LD-SBR conformance test procedure

The essential modules are:

- **Twiddle-estimation:** This module identifies the twiddles used in the synthesis filter bank in the codec under test, and based on this computes the analysis filterbank twiddles in the codec under test. This information is passed on to the reference decoder where the QMF filter bank is given the same distribution of the twiddle factors as used in the decoder under test.

- **Reference decoder** is a reference decoder according to the ISO specification. It generates a reference signal based on the stored low-band signal and the compressed MPEG-4 LD-SBR data.

- **Comparison test,** this module calculates the difference signals between the output from the decoder under test and the internal reference. The maximum amplitude of the difference signal as well as the RMS of the difference signal are calculated. These conformance criteria are specified with respect to PCM-sample in the range –32768 .. 32767.

The test procedure criteria specified in Table 77 — ELD with LD-SBR test sequences have to be applied. The testing shall be done using the test procedure defined above. The conformance of the underlying AAC ELD core shall be tested before conformance testing is done for the LD-SBR Tool.

Notes: The reference software implementation of the conformance test procedure will behave differently for the test sequences named er_eld_sbr_twi_*, since for these test sequences the QMF implementation is identified with respect to the twiddle factors used on the analysis and synthesis side of the QMF.

7.25.2.2 Test sequences

Table 77 — ELD with LD-SBR test sequences

file base name	Content	bitrate (kbit/s)	AudioObjectType	SamplingFrequencyIndex	ChannelConfiguration	QMF Identification	QMF Accuracy	Envelope Adjuster Accuracy	Grid control tests	Header Change Tests	Inverse Filtering Tests	Additional Sines Tests	Diff max	RMS max (linear value)	test procedure
er_eld_sbr_twi	None	48	39	4,7	1	y	y	-	-	-	-	y	-	-	-
er_eld_sbr_e	rectangle * 10Hz sine	48	39	5,6,7	1,2	-	-	Y	-	-	-	-	TBD	TBD	maxDiff/RMS
er_eld_sbr_gh	rectangle * 10Hz sine	48	39	5,6,7	1,2	-	-	-	y	y	-	-	TBD	TBD	maxDiff/RMS
er_eld_sbr_i	rectangle + noise	48	39	5,6,7	1,2	-	-	-	-	-	y	-	TBD	TBD	maxDiff/RMS
er_eld_sbr_s	Noise	48	39	5,6,7	1,2	-	-	-	-	-	-	y	TBD	TBD	maxDiff/RMS

8 Audio EP tool

8.1 Compressed data

8.1.1 Characteristics

Encoders may apply restrictions to the following parameters of the compressed data[1]:

8.1.1.1 AudioSpecificConfig

number_of_predefined_set

interleave_type

bit_stuffing

number_of_concatenated_frame

number_of_class

length_escape

rate_escape

crc_len_escape

concatenate_flag

fec_type

[1] With respect to the AudioSpecificConfig, only parameters of ErrorProtectionSpecificConfig are mentioned. For all other parameters please refer to the appropriate Clause 7.

termination_switch

interleave_switch

class_optional

number_of_bits_for_length

class_length

class_rate

class_crclen

class_reordered_output

class_output_order

header_protection

header_rate

header_crclen

rs_fec_capability

8.1.1.2 Bitstream payload

None.

8.1.2 Test procedure

Each compressed data shall meet the syntactic and semantic requirements specified in ISO/IEC 14496-3. This Subclause describes a set of semantic tests to be performed on decoder relevant data. The procedure to verify whether the syntax is correct is straightforward and therefore not defined in this Subclause. In the description of the semantic tests it is assumed that the tested compressed data contains no errors due to transmission or other causes. For each test the condition or conditions that must be satisfied are given, as well as the prerequisites or conditions in which the test can be applied.

8.1.2.1 AudioSpecificConfig

number_of_predefined_set: Shall not be encoded with 0.

interleave_type: Shall not be encoded with 3.

bit_stuffing: No restrictions apply.

number_of_concatenated_frame: No restrictions apply, as long as no escape mechanism is used. Otherwise number_of_concatenate_frame shall be 1.

number_of_class: No restrictions apply.

length_escape: No restrictions apply, as long as (number_of_concatenated_frame == 1). Otherwise length_escape shall be 0.

rate_escape: No restrictions apply, as long as (number_of_concatenated_frame == 1). Otherwise rate_escape shall be 0.

crc_escape: No restrictions apply, as long as (number_of_concatenated_frame == 1). Otherwise crc_escape shall be 0.

concatenate_flag: No restrictions apply.

fec_type: Shall not be encoded with 3. Considering a class with (fec_type == 2), the following class shall provide either (fec_type == 2) or (fec_type == 1), but not (fec_type == 0).

termination_switch: No restrictions apply.

interleave_switch: The same value shall be used for all classes that are protected by the same RS code (indicated by fec_type). No further restrictions apply.

class_optional: No restrictions apply.

number_of_bits_for_length: No restrictions apply.

class_length: No restrictions apply.

class_rate: Shall be less than 24 if (fec_type == 0). Otherwise no restrictions apply.

class_crclen: Shall be in the range of 0 and 18.

class_output_order: shall be less than number_of_class[i][j] element. Each value in the range of 0 and number_of_class[i][j] shall occur exacly on times.

header_protection: No restrictions apply.

header_rate: shall be less than 24 if (fec_type == 0). Otherwise no restrictions apply.

header_crclen: Shall be in the range of 0 and 18.

rs_fec_capability: No restrictions apply.

8.1.2.2 Bitstream payload

rs_ep_frame(): The bit number shall be less than (1+ mux. Redundancy caused by FEC/100) x max. frame length in the object including the profile and level.

rs_parity_bits: No restrictions apply.

interleaved_frame_mode1: No restrictions apply.

interleaved_frame_mode2: No restrictions apply.

stuffing_bits: No restrictions apply.

choice_of_pred: No restrictions apply.

choice_of_pred_parity: No restrictions apply.

class_attrib_parity: No restrictions apply.

class_bit_count[j]: No restrictions apply.

class_code_rate[j]: No restrictions apply.

class_crc_count[j]: No restrictions apply.

num_stuffing_bits: No restrictions apply.

ep_encoded_class[j]: No restrictions apply.

8.2 Decoders

8.2.1 Characteristics

A conforming decoder may also support any of the following modifications to the parameters in an audio compressed data:

Table 78 — EP tool parameter

Compressed data Characteristic	Variation
Redundancy	a decoder may support additional FEC beyond the minimums listed for its profile and level
# stages of interleaving	a decoder may support additional # stages of interleaving beyond the minimums listed for its profile and level

8.2.2 Test procedure

To test EP decoders, ISO/IEC JTC 1/SC 29/WG 11 supplies a number of test sequences. Supplied sequences cover all object types defined in ISO/IEC 14496-3. Compressed data with (epConfig == 2 || epConfig == 3) is provided for a subset of those test sequences used to test the individual audio object types. The test sequences are listed in Table 79.

To be called an ISO/IEC 14496 EP decoder the same conformance criteria as described for the individual audio object types have to be fulfilled. Furthermore, as already mentioned in Clause 4: The output of a conforming decoder shall be similar independent of the used epConfig setting.

Table 79 — EP tool test sequences

coreSetup	epSetup	layer	epConfig	directMaping	SamplingFrequencyIndex (if not specified: same as for epConfig=0,1)	number_of_predefinition_set	interleave_type	bit_stuffing	number_of_concatenated_frame	number_of_class	length_escape	rate_escape	crc_len_escape	concatenated_flag	fec_type	termination_switch	interleave_switch	class_optional	number_of_bits_for_length	class_length	class_rate	class_crclen	class_reordered_output	class_output_order	header_protection	header_rate	header_crclen
er_al10	crc01	-	3	1	11/7	1	0	1	1	3	1,1,1	0,0,0	0,0,0	-	1,0,1	-,0,-	-	0,1,0	10,10,14	-	0,0,0	1,2,0	0	-	0	-	-
er_al18	crc02	-	3	1	10/6	1	2	1	1	4	1,1,1,1	0,0,0,0	0,0,0,0	-	0,1,0,1	0,-,0,-	0,0,0,0	0,0,1,0	10,10,10,14	-	0,0,0,0	3,0,4,0	0	-	1	24	6
er_al21	crc03	-	3	1	9/5	1	2	1	1	7	1,1,1,1, 1,1,1	0,0,0,0 0,0,0	0,0,0,0, 0,0,0	-	0,1,0,1, 0,1,0	0,-,0,-, 0,-,0	0,0,0,0, 0,0,0	0,0,1,0, 0,1,0	8,10,10,14, 10,10,0	-	0,0,0,0, 0,0,0	5,6,7,0, 8,9,0	0	-	0	-	-
er_al23	crc04	-	3	1	8/4	1	0	1	1	9	1,1,1,1,1, 1,1,1,1	0,0,0,0,0, 0,0,0,0	0,0,0,0,0, 0,0,0,0	-	1,0,1,0,1, 0,1,0,1,0	-,0,-,0,-, -,0,-,0	-	0,0,0,1,0, 0,0,1,0	8,10,10,10,14, 10,10,10,0	-	0,0,0,0,0, 0,0,0,0	10,11,0,12,0 13,0,14,0	0	-	0	-	-
er_al15	srcpc01	-	3	1	11/3	1	2	1	1	3	1,1,1,1	0,0,0,0	0,0,0,0	-	0,0,0,0	0,0,0,0	2,0,1	0,1,0	10,10,0	-	8,0,0	15,16,0	0	-	0	-	-
er_al18	srcpc02	-	3	1	10/2	1	2	1	1	4	1,1,1,1, 1,1,1,1	0,0,0,0, 0,0,0,0	0,0,0,0, 0,0,0,0	-	0,0,0,0, 0,0,0,0	0,0,0,0, 0,0,0,0	2,1,0,1	0,0,1,0	10,10,10,0	-	8,0,0,0	17,0,18,0	0	-	0	-	-
er_al26	srcpc03	-	3	1	9/1	1	2	1	1	7	1,1,1,1, 1,1,1	0,0,0,0, 0,0,0	0,0,0,0, 0,0,0	-	0,0,0,0, 0,0,0	1,1,0,0, 1,0,0,0	2,2,0,3, 2,0,1	0,0,1,0, 0,1,0	8,10,10,14, 10,10,0	-	11,10,0,0, 9,0,0	9,8,7,0, 6,5,0	0	-	1	17	5
er_ap27	srcpc04	-	3	1	8/0	1	2	1	1	9	1,1,1,1,1, 1,1,1,1	0,0,0,0,0, 0,0,0,0	0,0,0,0,0, 0,0,0,0	-	0,0,0,0,0, 0,0,0,0	1,0,0,0, 0,0,0,0	2,2,1,0,3, 2,1,0,1	0,0,0,1,0, 0,0,1,0	8,10,10,10,14, 10,10,10,0	-	7,6,5,0,0, 4,3,0,0	18,18,18,13,1, 15,16,17,2	0	-	0	-	-
er_ad115	srs01	-	3	1	7/4	1	2	1	1	3	1,1,1	0,0,0	0,0,0	-	1,0,0	-,0,0	1,0,1	0,1,0	10,10,0	-	8,0,0	15,16,0	0	-	0	-	-
er_ap14	srs02	-	3	1	9/5	1	2	1	1	4	1,1,1,1	0,0,0,0	0,0,0,0	-	1,0,0,0	-,1,1,0	0,0,1,0,1	0,0,1,0	10,10,10,0	-	8,0,0,0	17,0,18,0	0	-	0	-	-
er_ad206	srs03	-	3	1	6/3	1	2	1	1	7	1,1,1,1, 1,1,1	0,0,0,0, 0,0,0	0,0,0,0, 0,0,0	-	2,2,0,0, 1,0,0	-,-,0,0, -,0,0	1,1,0,3, 1,0,1	0,0,1,0, 0,1,0	8,10,10,14, 10,10,0	-	9,9,0,0, 9,0,0	9,8,7,0, 6,5,0	1	0,1,4, 2,5,3,6	0	-	-
er_ad218	srs04	-	3	1	5	1	1	1	1	9	1,1,1,1,1, 1,1,1,1	0,0,0,0,0, 0,0,0,0	0,0,0,0,0, 0,0,0,0	-	2,2,0,0,0, 1,0,0,0	-,-,0,0,0, -,0,0,0	-	0,0,0,1,0, 0,0,1,0	8,10,10,10,14, 10,10,10,0	-	20,20,0,0,0, 20,0,0,0	18,18,18,13,1, 15,16,17,2	1	0,1,5,2, 6,3,7,4, 8	1	19	7
er_ac111	cplx01	0	3	1	-	1	0	1	1	3	1,1,1	0,0,0	0,0,0	-	0,0,0	1,1,1	-	0,1,0	10,10,0	-	24,23,0	1,2,3	0	-	0	-	-
		1								7	1,1,1,1, 1,1,1	0,0,0,0, 0,0,0	0,0,0,0, 0,0,0	-	2,2,0,0, 1,0,0	-,-,1,0, -,0,1,	-	0,0,1,0, 0,1,0	8,10,10,14, 10,10,0	-	31,31,21,0, 31,19,0	4,5,6,7, 8,9,10	1	0,1,4, 2,5,3,6	0	-	-
er_ac123	cplx02	0	3	1	-	1	0	1	1	9	1,1,1,1,1, 1,1,1,1	0,0,0,0,0, 0,0,0,0	0,0,0,0,0, 0,0,0,0	-	0,0,0,0,0, 0,0,0,0	0,0,0,0,0, 0,0,0,0	-	0,0,0,1,0, 0,0,1,0	8,10,10,10,14, 10,10,10,0	-	15,14,13,12,7, 10,9,11,2	11,12,13,14,15 16,17,18,17	1	0,1,5,2, 6,3,7,4, 8	0	-	-
		1	3	1	-	1	1	1	1	9	1,1,1,1,1, 1,1,1,1	0,0,0,0,0, 0,0,0,0	0,0,0,0,0, 0,0,0,0	-	1,1,1,1,1, 1,1,1,1	-,-,-,-,-, -,-,-,-	-	0,0,0,1,0, 0,0,1,0	8,10,10,10,14, 10,10,10,0	-	0,1,2,3,4 5,6,7,8	16,15,14,13,12, 11,10,9,8	0	-	0	-	-
er_ac211	cplx03	0	3	1	-	1	1	1	1	5	0,0,0,0,0	0,0,0,0,0	0,0,0,0,0	-	0,0,0,0,0	0,0,0,0,0	-	0,0,0,0,0	-	4,9,12,31,64	7,5,3,1,0	8,6,4,2,0	0	-	0	-	-
		1	3	1	-	1	2	1	1	4	1,1,1,1	0,0,0,0	0,0,0,0	-	2,2,2,1	-,-,-,-	0,0,0,0	0,0,1,0	10,10,10,14	-	22,22,22,22	7,6,5,4	0	-	0	-	-
		2	3	1	-	1	0	1	1	7	1,1,1,1, 1,1,1	0,0,0,0, 0,0,0	0,0,0,0, 0,0,0	-	2,2,2,2, 2,2,1	-,-,-,-, -,-,-	-	0,0,1,0, 0,1,0	8,10,10,14, 10,10,14	-	0,0,0,0, 0,0,0	0,1,2,3, 4,5,6	1	0,1,4, 2,5,3,6	0	-	-
er_ac221	cplx04	0	3	1	-	1	1	1	1	5	0,0,0,0,0	0,0,0,0,0	0,0,0,0,0	-	1,1,1,1,1	-,-,-,-,-	-	0,0,0,0,0	-	6,13,20,49,98	7,5,3,1,0	9,7,5,3,1	0	-	0	-	-
		1	3	1	-	1	1	1	1	7	1,1,1,1, 1,1,1	0,0,0,0, 0,0,0	0,0,0,0, 0,0,0	-	0,1,0,1, 0,1,0	0,-,1,-, 1,-,0	-	0,0,0,1, 0,1,0	8,10,10,14, 10,10,0	-	0,0,0,1,1, 22,3,4	7,8,9,10, 13,14,15	1	0,1,4, 2,5,3,6	0	-	-
		2	3	1	-	1	2	1	1	9	1,1,1,1,1, 1,1,1,1	0,0,0,0,0, 0,0,0,0	0,0,0,0,0, 0,0,0,0	-	2,2,1,1,1, 1,1,1,1	-,-,-,-,-, -,-,-,-	1,1,1,3,0, 1,3,1,0	0,0,0,1,0, 0,0,1,0	8,10,10,10,14, 10,10,10,14	-	0,0,0,0,0, 0,0,0,0	16,17,18,17,16, 15,14,13,12	1	0,1,5,2, 6,3,7,4, 8	0	-	-

coreSetup	epSetup	layer	epConfig	directMaping	SamplingFrequencyIndex (if not specified: same as for epConfig=0,1)	number_of_predefinition_set	interleave_type	bit_stuffing	number_of_concatenated_frame	number_of_class	length_escape	rate_escape	crc_len_escape	concatenated_flag	fec_type	termination_switch	interleave_switch	class_optional	number_of_bits_for_length	class_length	class_rate	class_crclen	class_reordered_output	class_output_order	header_protection	header_rate	header_crclen
er_ac311	cplx05	0	2	1	5	1	0	1	1	2	0,0	0,0	0,0	-	0,0	0,0	-	0,0	-	64,192	0,0	8,0	0	-	0	-	-
		1	3	1	5	1	0	1	1	3	1,1,1	0,0,0	0,0,0	-	2,1,0	-,-,0	-	0,1,0	10,10,0	-	8,8,5	11,10,9	0	-	0	-	-
		2	3	1	5	1	1	1	1	9	1,1,1,1,1, 1,1,1,1	0,0,0,0,0, 0,0,0,0	0,0,0,0,0, 0,0,0,0	-	1,2,1,1,1, 1,2,2,1	-,-,-,-,-, -,-,-,-	-	0,0,0,1,0, 0,0,1,0	8,10,10,10,14, 10,10,10,14	-	11,11,17,1,0, 19,19,19,21	8,7,6,5,0, 1,2,3,4	1	8,7,6,5, 4,3,2,1, 0	0	-	-
er_ac321	cplx06	0	2	1	6	1	1	1	1	2	0,0	0,0	0,0	-	0,0	0,0	-	0,0	-	80,400	6,6	8,8	0	-	0	-	-
		1	3	1	6	1	2	1	1	9	1,1,1,1,1, 1,1,1,1	0,0,0,0,0, 0,0,0,0	0,0,0,0,0, 0,0,0,0	-	0,1,2,1,0, 0,1,1,0	1,-,-,-,1, 0,-,-,0	3,3,1,1,0, 1,1,3,2	0,0,0,1,0, 0,0,1,0	8,10,10,10,14, 10,10,10,0	-	6,6,1,1,1, 8,3,3,0	5,6,7,8,9, 10,11,12,13	0	-	0	-	-
		2	3	1	6	1	0	1	1	7	1,1,1,1, 1,1,1	0,0,0,0, 0,0,0	0,0,0,0, 0,0,0	-	2,1,1,2, 2,2,1	-,-,-,-, -,-,-	-	0,0,1,0, 0,1,0	8,10,10,14, 10,10,14	-	24,24,12,6, 12,6,6	14,15,16,17, 18,6,0	1	0,1,4, 2,5,3,6	0	-	-
er_tv01		-	2	-	5	1	0	1	1	2	0,0	0,0	0,0	-	0,0	0,0	-	0,0	-	128,896	0,0	8,0	0	-	0	-	-
er_tv02		0	2	-	3	1	1	1	1	2	0,0	0,0	0,0	-	0,0	0,0	-	0,0	-	136,200	6,0	8,0	0	-	0	-	-
.		1	2	-	3	1	1	1	1	2	0,0	0,0	0,0	-	0,0	0,0	-	0,0	-	64,272	4,0	8,0	0	-	0	-	-
.		2	2	-	3	1	1	1	1	2	0,0	0,0	0,0	-	0,0	0,0	-	0,0	-	64,272	2,0	8,0	0	-	0	-	-
er_ce00			3	-	-	4	0	1	1	1 / 5 / 1 / 1	0 / 0,0,0,0,0 / 0 / 0	0 / 0,0,0,0,0 / 0 / 0	0 / 0,0,0,0,0 / 0 / 0	-	0 / 0,0,0,0,0 / 0 / 0	0 / 0,0,0,0,0 / 0 / 0	-	0 / 0,0,0,0,0 / 0 / 0	-	2 / 8,13,20,37,78 / 23 / 8	8 / 8,6,4,2,0 / 8 / 8	5 / 5,6,7,8,0 / 5 / 5	0	-	0	-	-
er_ce02		-	3	-	-	1	1	1	1	5	0,0,0,0,0	0,0,0,0,0	0,0,0,0,0	-	0,0,0,0,0	0,0,0,0,0	-	0,0,0,0,0	-	4,9,12,31,64	8,6,4,2,0	5,6,7,8,0	0	-	0	-	-
er_ce06		0	3	-	-	4	0	1	1	1 / 5 / 1 / 1	0 / 0,0,0,0,0 / 0 / 0	0 / 0,0,0,0,0 / 0 / 0	0 / 0,0,0,0,0 / 0 / 0	-	0 / 0,0,0,0,0 / 0 / 0	0 / 0,0,0,0,0 / 0 / 0	-	0 / 0,0,0,0,0 / 0 / 0	-	2 / 7,11,16,39,85 / 23 / 8	8 / 8,6,4,2,0 / 8 / 8	5 / 5,6,7,8,0 / 5 / 5	0	-	0	-	-
		1	3	-	-	2	0	1	1	0 / 1	- / 0	- / 0	- / 0	-	- / 0	- / 0	-	- / 0	-	0 / 60	- / 0	- / 8	0	-	0	-	-
		2	3	-	-	2	0	1	1	0 / 1	- / 0	- / 0	- / 0	-	- / 0	- / 0	-	- / 0	-	0 / 60	- / 0	- / 8	0	-	0	-	-
		3	3	-	-	3	0	1	1	0 / 1 / 1	- / 0 / 0	- / 0 / 0	- / 0 / 0	-	- / 0 / 0	- / 0 / 0	-	- / 0 / 0	-	0 / 320 / 21	- / 0 / 0	- / 10 / 5	0	-	0	-	-
er_ce09		-	3	-	-	5	0	1	1	1 / 5 / 5 / 1 / 1	0 / 0,0,0,0,0 / 0,0,0,0,0 / 0 / 0	0 / 0,0,0,0,0 / 0,0,0,0,0 / 0 / 0	0 / 0,0,0,0,0 / 0,0,0,0,0 / 0 / 0	-	0 / 0,0,0,0,0 / 0,0,0,0,0 / 0 / 0	0 / 0,0,0,0,0 / 0,0,0,0,0 / 0 / 0	-	0 / 0,0,0,0,0 / 0,0,0,0,0 / 0 / 0	-	2 / 21,14,19,39,7 7 / 6,8,9,26,75 / 40 / 8	8 / 8,6,4,2,0 / 8,6,4,2,0 / 8 / 8	5 / 5,6,7,8,0 / 5,6,7,8,0 / 5 / 5	0	-	0	-	-

coreSetup	epSetup	layer	epConfig	directMaping	SamplingFrequencyIndex (if not specified: same as for epConfig=0,1)	number_of_predefinition_set	interleave_type	bit_stuffing	number_of_concatenated_frame	number_of_class	length_escape	rate_escape	crc_len_escape	concatenated_flag	fec_type	termination_switch	interleave_switch	class_optional	number_of_bits_for_length	class_length	class_rate	class_crclen	class_reordered_output	class_output_order	header_protection	header_rate	header_crclen
er_ce13		-	3	-	-	2	1	1	1	5 / 5	0,0,0,0,0 / 0,0,0,0,0	0,0,0,0,0 / 0,0,0,0,0	0,0,0,0,0 / 0,0,0,0,0	-	0,0,0,0,0 / 0,0,0,0,0	0,0,0,0,0 / 0,0,0,0,0	-	0,0,0,0,0 / 0,0,0,0,0		34,18,26,21,6 3 / 19,12,16,8,61	8,6,4,2,0 / 8,6,4,2,0	5,6,7,8,0 / 5,6,7,8,0	0	-	0	-	-
		-	3	-	-	2	1	1	2	1 / 1	0,0,0,0,0 / 0,0,0,0,0	0,0,0,0,0 / 0,0,0,0,0	0,0,0,0,0 / 0,0,0,0,0	-	0,0,0,0,0 / 0,0,0,0,0	0,0,0,0,0 / 0,0,0,0,0	-	0,0,0,0,0 / 0,0,0,0,0		2 / 50,30,42,29,1 31 / 40 / 8	8 / 8,6,4,2,0 / 8	5 / 5,6,7,8,0 / 5	0	-	0	-	-
		-	3	-	-	5	0	1	1	1/ 5 / 5 / 1 / 1	0,0,0,0,0 / 0,0,0,0,0 / 0,0,0,0,0 / 0 / 0	0,0,0,0,0 / 0,0,0,0,0 / 0,0,0,0,0 / 0 / 0	0,0,0,0,0 / 0,0,0,0,0 / 0,0,0,0,0 / 0 / 0	-	0,0,0,0,0 / 0,0,0,0,0 / 0,0,0,0,0 / 0 / 0	0,0,0,0,0 / 0,0,0,0,0 / 0,0,0,0,0 / 0 / 0	-	0,0,0,0,0 / 0,0,0,0,0 / 0,0,0,0,0 / 0 / 0		2 / 60,36,50,33,1 63 / 45,30,40,20,1 61 / 40 / 8	8 / 8,6,4,2,0 / 8,6,4,2,0 / 8	5 / 5,6,7,8,0 / 5,6,7,8,0 / 5	0	-	0	-	-
er_hv03	crc	-	2	-	-	1	0	1	2	4	0,0,0,0	0,0,0,0	0,0,0,0	1,0,0 ,1	0,0,0,0	0,0,0,0	-	0,0,0,0		22,4,4,10	0,0,0,0	6,1,1,0	0	-	0	-	-
er_hv04	crc	-	2	-	-	1	0	1	2	5	0,0,0,0,0	0,0,0,0,0	0,0,0,0,0	1,1,0,0, 1	0,0,0,0,0	0,0,0,0,0	-	0,0,0,0,0		33,22,4,4,17	0,0,0,0,0	6,6,1,1,0	0	-	0	-	-
er_hv03	fec	-	2	-	-	1	2	1	2	4	0,0,0,0	0,0,0,0	0,0,0,0	1,0,0,1	0,0,0,0	0,0,0,0	2,3,3,3	0,0,0,0		22,4,4,10	8,0,0,0	6,1,1,0	0	-	0	-	-
er_hv04	fec	-	2	-	-	1	2	1	2	5	0,0,0,0,0	0,0,0,0,0	0,0,0,0,0	1,1,0,0, 1	0,0,0,0,0	0,0,0,0,0	2,3,3,3,3	0,0,0,0,0		33,22,4,4,17	8,0,0,0,0	6,6,1,1,0	0	-	0	-	-
er_bs01		-	3	1	-	1	1	1	1	6	1,1,1,1,1,1	0,0,0,0,0,0	0,0,0,0,0,0	-	0,0,0,0,0,0	0,0,0,0,0,0	-	0,0,0,0,0,0	7,11,10,10,10, 0	-	20,16,8, 0,0,0	7,10,10, 10,10,10	0	-	0	-	-
er_bs02		-	3	1	-	1	1	1	1	6	1,1,1,1,1,1	0,0,0,0,0,0	0,0,0,0,0,0	-	0,0,0,0,0,0	0,0,0,0,0,0	-	0,0,0,0,0,0	7,11,10,10,10, 0	-	20,16,8, 0,0,0	7,10,10, 10,10,10	0	-	0	-	-
er_bs03		0	3	1	-	1	1	1	1	2	1,1	0,0	0,0	-	0,0	0,0	-	0,0	8,0	-	20,16	7,10	0	-	0	-	-
		1	3	1	-	1	1	1	1	4	1,1,1,1	0,0,0,0	0,0,0,0	-	0,0,0,0	0,0,0,0	-	0,0,0,0	11,11,11,0	-	8,0,0,0	10,10,10,10	0	-	0	-	-
er_bs04		0	3	1	-	1	1	1	1	2	1,1	0,0	0,0	-	0,0	0,0	-	0,0	10,0	-	20,16	7,10	0	-	0	-	-
		1	3	1	-	1	1	1	2	2	1,1	0,0	0,0	-	0,0	0,0	-	0,0	11,0	-	8,0	10,10	0	-	0	-	-
		2	3	1	-	1	1	1	1	1	1	0	0	-	0	0	-	0	0	-	0	10	0	-	0	-	-
		3	3	1	-	1	1	1	1	1	1	0	0	-	0	0	-	0	0	-	0	10	0	-	0	-	-
er_bs05		0	3	1	-	1	1	1	1	2	1,1	0,0	0,0	-	0,0	0,0	-	0,0	10,0	-	20,16	7,10	0	-	0	-	-
		1	3	1	-	1	1	1	1	1	1	0	0	-	0	0	-	0	0	-	8	10	0	-	0	-	-
		2	3	1	-	1	1	1	1	1	1	0	0	-	0	0	-	0	0	-	0	10	0	-	0	-	-
		3	3	1	-	1	1	1	1	1	1	0	0	-	0	0	-	0	0	-	0	10	0	-	0	-	-
		4	3	1	-	1	1	1	1	1	1	0	0	-	0	0	-	0	0	-	0	10	0	-	0	-	-

coreSetup	epSetup	layer	epConfig	directMaping	SamplingFrequencyIndex (if not specified: same as for epConfig=0,1)	number_of_predefinition_set	interleave_type	bit_stuffing	number_of_concatenated_frame	number_of_class	length_escape	rate_escape	crc_len_escape	concatenated_flag	fec_type	termination_switch	interleave_switch	class_optional	number_of_bits_for_length	class_length	class_rate	class_crclen	class_reordered_output	class_output_order	header_protection	header_rate	header_crclen
er_bs06		0	3	1	-	1	1	1	1	2	1,1	0,0	0,0	-	0,0	0,0	-	0,0	10,0	-	20,16	7,10	0	-	0	-	-
		1	3	1	-	1	1	1	1	2	1,1	0,0	0,0	-	0,0	0,0	-	0,0	11,0	-	8,0	10,10	0	-	0	-	-
er_bs07		2	3	1	-	1	1	1	1	2	1,1	0,0	0,0	-	0,0	0,0	-	0,0	11,0	-	0,0	10,10	0	-	0	-	-
		-	3	1	-	1	1	1	1	6	1,1,1,1,1,1	0,0,0,0,0,0	0,0,0,0,0,0	-	0,0,0,0,0,0	0,0,0,0,0,0	-	0,0,0,0,0,0	7,11,10,10,10,0	-	20,16,8,0,0,0	7,10,10,10,10,1,0	0	-	0	-	-
er_bs08		-	3	1	-	1	1		1	6	1,1,1,1,1,1	0,0,0,0,0,0	0,0,0,0,0,0	-	0,0,0,0,0,0	0,0,0,0,0,0	-	0,0,0,0,0,0	7,11,10,10,10,0	-	20,16,8,0,0,0	7,10,10,10,10,1,0	0	-	0	-	-
er_bs09		-	3	1	-	1	1	1	1	6	1,1,1,1,1,1	0,0,0,0,0,0	0,0,0,0,0,0	-	0,0,0,0,0,0	0,0,0,0,0,0	-	0,0,0,0,0,0	10,14,9,13,10,0	-	20,0,16,0,8,0	10,10,9,10,10,10	0	-	0	-	-
er_bs10		-	3	1	-	1	1	1	1	4	1,1,1,1	0,0,0,0	0,0,0,0	-	0,0,0,0	0,0,0,0	-	0,0,0,0	10,14,9,0	-	20,0,16,0	10,10,9,10	0	-	0	-	-
er_bs11		-	3	1	-	1	1	1	1	9	1,1,1,1,1,1,1,1,1	0,0,0,0,0,0,0,0,0	0,0,0,0,0,0,0,0,0	-	0,0,0,0,0,0,0,0,0	0,0,0,0,0,0,0,0,0	-	0,0,0,0,0,0,0,0,0	10,14,9,13,9,10,13,10,0	-	20,0,16,0,0,8,0,8,0	10,10,9,10,10,10,8,10,10,10	0	-	0	-	-
er_hi26	crc	-	3	1	-	1	0	1	1	5	1,1,1,1,1	0,0,0,0,0	0,0,0,0,0	-	0,0,0,0,0	0,0,0,0,0	-	0,0,0,0,1	6,8,12,12,0	-	0,0,0,0,0	6,9,12,12,4	0	-	0	-	-
er_hi27	crc	-	3	1	-	1	0	1	1	5	1,1,1,1,1	0,0,0,0,0	0,0,0,0,0	-	0,0,0,0,0	0,0,0,0,0	-	0,0,0,0,1	6,8,12,12,0	-	0,0,0,0,0	6,9,12,12,4	0	-	0	-	-
er_hi28	crc	-	3	1	-	1	0	1	1	5	1,1,1,1,1	0,0,0,0,0	0,0,0,0,0	-	0,0,0,0,0	0,0,0,0,0	-	0,0,0,0,1	6,8,12,12,0	-	0,0,0,0,0	6,9,12,12,4	0	-	0	-	-
er_hi29	crc	-	3	1	-	1	0	1	1	5	1,1,1,1,1	0,0,0,0,0	0,0,0,0,0	-	0,0,0,0,0	0,0,0,0,0	-	0,0,0,0,1	6,8,12,12,0	-	0,0,0,0,0	6,9,12,12,4	0	-	0	-	-
er_hi26	fec	-	3	1	-	1	1	1	1	5	1,1,1,1,1	0,0,0,0,0	0,0,0,0,0	-	0,0,0,0,0	0,0,0,0,0	-	0,0,0,0,1	6,8,12,12,0	-	8,4,2,0,0	6,9,12,12,4	0	-	0	-	-
er_hi27	fec	-	3	1	-	1	1	1	1	5	1,1,1,1,1	0,0,0,0,0	0,0,0,0,0	-	0,0,0,0,0	0,0,0,0,0	-	0,0,0,0,1	6,8,12,12,0	-	8,4,2,0,0	6,9,12,12,4	0	-	0	-	-
er_hi28	fec	-	3	1	-	1	1	1	1	5	1,1,1,1,1	0,0,0,0,0	0,0,0,0,0	-	0,0,0,0,0	0,0,0,0,0	-	0,0,0,0,1	6,8,12,12,0	-	8,4,2,0,0	6,9,12,12,4	0	-	0	-	-
er_hi29	fec	-	3	1	-	1	1	1	1	5	1,1,1,1,1	0,0,0,0,0	0,0,0,0,0	-	0,0,0,0,0	0,0,0,0,0	-	0,0,0,0,1	6,8,12,12,0	-	8,4,2,0,0	6,9,12,12,4	0	-	0	-	-

9 Audio Composition

9.1 AudioBIFS v1

9.1.1 Introduction

This Clause defines the conformance of audio composition using AudioBIFS nodes as defined in ISO/IEC 14496-11 (Scene description and application engine). The nodes that are related to audio composition in BIFS are: AudioSource, Sound, Sound2D, AudioClip, AudioBuffer, AudioFX, AudioMix, AudioSwitch, AudioDelay, Transform2D, Transform3D and Listening Point. Nodes that have conformance points have to be tested with the Null Object Type *or* the output of one of the decoders as defined in the following. The CELP decoder shall be used for testing Speech and Scalable Audio Profiles. The Structured Audio decoder shall be used for testing the Synthetic and Main Audio Profiles. At least three identifiable test signals per decoder are needed in order to be able to test the functionality of some nodes (e.g. AudioSwitch, AudioMix).

9.1.1.1 Complexity issues in AudioBIFS nodes

The following parameters have been identified to bound audio composition complexity. The table below gives an overview of possible restrictions:

Table 80 — BIFS complexity restrictive parameters

Audio Feature	Restrictive parameters	Remarks
BIFS Field Update	Maximum reaction time, until a BIFS field update is achieved	
AudioMix, AudioSwitch, AudioSource	Maximum width, maximum depth of the sub-tree, click-free switching	
AudioDelay, AudioClip, AudioBuffer	Total buffer memory, click-free delay	
Sample Rate Conversion	Total conversion processing power, sample-rate conversion ratios.	
AudioFX	According to the restrictions of SA approved by the Audio group.	According to SAOL level definition based on the complexity metrics.
Sound, Sound2D	#spatialized	

Parameter definitions:

- Depth of an audio sub-tree: maximum number of consecutive nodes from the output of a AudioSource or AudioClip node to the input of a Sound/Sound2D node.
- Width of audio sub-tree: maximum number of parallel channels from the output of an AudioSource or AudioClip node to the input of a Sound/Sound2D node.
- Total Memory Buffer: an amount of memory needed to store samples shared between the different AudioDelay, AudioClip and AudioBuffer nodes present in a scene according to the formula:

$$Total\ Memory = \mathbf{SUM}(NbChannels(j) * NbBufferedSamples(j))$$

where: *j is the considered node,*
NbChannels is the number of channels for this node
*NbBufferedSamples = Delay(j)*SamplingFrequency(j)*

- Reaction Time of a BIFS field update is the maximum time in msec. until the changes is audible .

- Total Conversion Processing Power: an amount of PCU shared among the different sampling rate conversions present in a scene according to 1.5.2.2 of ISO/IEC 14496-3: Complexity Units
- Spatializable Object types: number of possible spatialized channels
- AudioFX: see 7.15
- Reaction Time of a BIFS field update: the maximum time in msec. until the changes is audible.

9.1.1.2 Levels for Systems Audio Scene Graph Profile

Following these considerations, audio composition Levels are defined in the form of the following table:

Table 81 — Systems Audio Scene Graph Profile Levels

Audio Parameter	Level 1	Level 2	Level 3	Level 4
Reaction time [msec]	64	32	32	16
Width	8	32	64	128
Depth	1	4	6	8
Click free fadings	N	Y	Y	HQ
Total memory buffer	256 ksamples	512 ksamples	2 Megasamples	6 Megasamples (2s for 64 channels at 48 kHz)
SR Conversion ratio	1	INT	any allowed ratio	any allowed ratio
Total Conversion Processing Power	0 (sampling rate conversion is forbidden)	16 PCU	64 PCU	128 PCU
AudioFX	Very Low Complexity (Table 64)	Low Complexity (Table 64)	Medium Complexity (Table 64)	High Complexity (Table 64)
Spatialization	0	4	16	32

9.1.1.3 Composition Unit Inputs

Table 82 — Composition Unit Inputs

Bitstream Type	**Bitstream Specification / Audio Profile**			
	Main	*Speech*	*Scalable*	*Synthetic*
Null Object Type (optional)	CU1_Px-CU4_Px	CU1_Px-CU4_Px	CU1_Px-CU4_Px	CU1_Px-CU4_Px
CELP decoder	-	CU1_Cx-CU4_Cx	CU1_Cx-CU4_Cx	-
SA decoder	CU1_Sx-CU4_Sx	-	-	CU1_Sx-CU4_Sx

The bitstream inputs are defined as follows:

CU1_Px Composition Unit Input PCM: 440 Hz sine, length 5 seconds + silence, length 1 second + 880 Hz sine, length 5 seconds + silence, length 1 second; sampling rate CU1_Pa 8 kHz, CU1_Pb 16 kHz, CU1_Pc 22.050 kHz

CU2_Px Composition Unit Input PCM: 440 Hz to 880 Hz linear sinesweep, length 5 seconds + silence, length 1 second + 440 Hz to 1760 Hz linear sinesweep, length 5 seconds + silence, length 1 second; sampling rate CU2_Pa 8 kHz, CU2_Pb 16 kHz, CU2_Pc 22.050 kHz

CU3_Px Composition Unit Input PCM: 440 Hz to 880 Hz logarithmic sinesweep, length 5 seconds + silence, length 1 second; sampling rate CU3_Pa 8 kHz, CU3_Pb 16 kHz, CU3_Pc 22.050 kHz

CU4_Px Composition Unit Input PCM: 440 Hz square wave, length 5 seconds + silence, length 1 second + 880 Hz square wave, length 5 seconds + silence, length 1 second, + 1760 Hz square wave, length 5 seconds + silence, length 1 second; sampling rate CU4_Pa 8 kHz, CU4_Pb 16 kHz, CU4_Pc 22.050 kHz

CU1_Cx Composition Unit Input CELP: 440 Hz sine, length 5 seconds + silence, length 1 second + 880 Hz sine, length 5 seconds + silence, length 1 second; sampling rate CU1_Ca 8 kHz, CU1_Cb 16 kHz

CU2_Cx Composition Unit Input CELP: 440 Hz to 880 Hz linear sinesweep, length 5 seconds + silence, length 1 second + 440 Hz to 1760 Hz linear sinesweep, length 5 seconds + silence, length 1 second; sampling rate CU2_Ca 8 kHz, CU2_Cb 16 kHz

CU3_Cx Composition Unit Input CELP: 440 Hz to 880 Hz logarithmic sinesweep, length 5 seconds + silence, length 1 second; sampling rate CU3_Ca 8 kHz, CU3_Cb 16 kHz

CU4_Cx Composition Unit Input CELP: 440 Hz square wave, length 5 seconds + silence, length 1 second + 880 Hz square wave, length 5 seconds + silence, length 1 second, + 1760 Hz square wave, length 5 seconds + silence, length 1 second; sampling rate CU4_Ca 8 kHz, CU4_Cb 16 kHz

CU1_Sx Composition Unit Input SA: 440 Hz sine, length 5 seconds + silence, length 1 second + 880 Hz sine, length 5 seconds + silence, length 1 second; sampling rate CU1_Sa 8 kHz, CU1_Sb 16 kHz, CU1_Sc 22.050 kHz

CU2_Sx Composition Unit Input SA: 440 Hz to 880 Hz linear sinesweep, length 5 seconds + silence, length 1 second + 440 Hz to 1760 Hz linear sinesweep, length 5 seconds + silence, length 1 second; sampling rate CU2_Sa 8 kHz, CU2_Sb 16 kHz, CU2_Sc 22.050 kHz

CU3_Sx Composition Unit Input SA: 440 Hz to 880 Hz logarithmic sinesweep, length 5 seconds + silence, length 1 second; sampling rate CU3_Sa 8 kHz, CU3_Sb 16 kHz, CU3_Sc 22.050 kHz

CU4_Sx Composition Unit Input SA: 440 Hz square wave, length 5 seconds + silence, length 1 second + 880 Hz square wave, length 5 seconds + silence, length 1 second, + 1760 Hz square wave, length 5 seconds + silence, length 1 second; sampling rate CU4_Sa 8 kHz, CU4_Sb 16 kHz, CU4_Sc 22.050 kHz

CELP composition units are encoded as in formats 0 and 14 of this standard, i.e. 8 kHz bitstreams are encoded with MPE, FRC set to off at 8300 bps, 16 kHz bitstreams are encoded with RPE, FRC set to on at 22000 bps.

Audio bitstreams for the defined composition units are provided by the electronic attachment to this part of ISO/IEC 14496.

9.1.1.4 Compositor Output

The output of the audio compositor will be investigated for conformance, and shall be a single output, *N* channel PCM audio stream The test CU sequences have a precision of 32 bits, but they can be converted to a precision (P) of 24 bits, where the most significant bit (MSB) will be labeled bit 0 and the least-significant bit (LSB) will be labeled bit 23. The output signal of the decoder under test is required to be in the same format. In the case that the output of the decoder has a precision of P' bits and if P' is smaller than 24, then the output is extended to 24 bits by setting bit P' through bit 23 to zero. In the next step, the difference (*diff*) of the samples of these signals has to be calculated. Every channel of a multichannel bitstream shall be tested. The total number of samples for each channel is N. A more precise description of the output format is 7.2.2.116 (Sound) and 7.2.2.117 (Sound2D) of ISO/IEC 14496-11.

Audio composition is tested for Conformance as described in the following 9.1.3 to 9.1.10. Test scenes are defined using composition units described in 9.1.1.3 with identifiers like CUN_Yx. N is a specified number from 1 to 4; Y and x, when not specified, shall be selected according to the Audio Profile@Level, Y as in 9.1.1.3, x according to the maximum sampling-rate supported by the same Audio Profile@Level.

EXAMPLE — CU2_Yx: in Main Profile this means CU2_Sc (Structured Audio at 22.050 kHz) and (optionally) CU2_Pc.

9.1.2 Common Audio Composition Characteristic

Common audio manipulations are operations that occur when presenting or modifying single or multiple elementary audio streams. Such operations are BIFS field changes, audio source switching, audio level changing, sample rate conversion etc.

9.1.2.1 BIFS field change reaction time

Audio node fields like pitch or speed in the AudioSource node or intensity or location in the Sound node may be changed interactively during the playback time. It is strongly recommended that these changes are audible at least 20 ms after the field has been changed. This time shall be measured from the instant when the change is detected by the MPEG-4 terminal until the instant when a change in the PCM output is measured.

9.1.2.2 Audio Switching and Level changes

Any hard switching or -level changing of audio signals will always cause perceived audible clicks and pops due to the broadband character of the step function. This effect may be tolerable in some low quality game applications, but is in general not acceptable.

One solution could be for implementations to smooth transitions by means of cross fade functions, which is common practice in professional audio workstations or digital mixing consoles. The duration is usually around (10..40) msec.

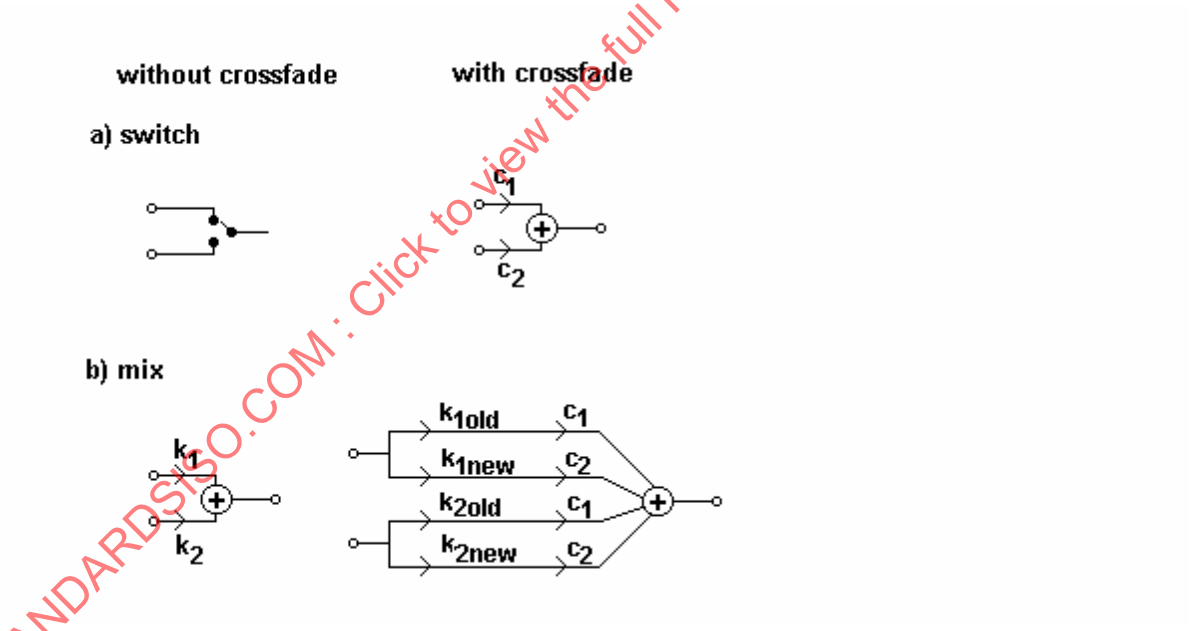

Figure 14 — Click free switch and mix

The above explained cross fade applies to the nodes **AudioSource**, **AudioMix**, **AudioSwitch**, and **AudioClip**.

9.1.2.3 Sample Rate Conversion

If the various children of a **Sound**/**Sound2D** node do not produce output at the same sampling rate, then the lengths of the output buffers of the children do not match, and the sampling rates of the children's' output must be brought into alignment in order to place their output buffers in the input buffer of the parent node. The sampling rate of the input buffer for the node shall be the fastest of the sampling rates of the children. The output buffers of the children shall be resampled to be at this sampling rate. The particular method of

resampling is non-normative, but the quality shall be close in accuracy to the DAC that the signal is targeted for, i.e. according to the rule dB SNR = 6 * (nbits -1), where nbits is the number of bits corresponding to the maximum bit depth of any of the signals being so converted and/or composited. Aliasing artifacts may be at this level of signal-to-noise ratio. The noise level due to arithmetic accuracy and other uncorrelated noise sources should be below the rule dB SNR = 6* nbits.

Content authors are advised that content which contains audio sources operating at many different sampling rates, especially sampling rates which are not related by simple rational values, may produce scenes with a high computational complexity.

The output sampling rate of a node shall be the output sampling rate of the input buffers after this resampling procedure is applied.

EXAMPLE — Suppose that node N has children M1 and M2, all three audio nodes, and that M1 and M2 produce output at S1 and S2 sampling rates respectively, where S1 > S2. Then if the decoding frame rate is F frames per second, then M1's output buffer will contain S1/F samples of data, and M2's output buffer will contain S2/F samples of data. Then, since M1 is the faster of the children, its output buffer values are placed in the input buffer of N. The output buffer of M2 is resampled by the factor S1/S2 to be S1/F samples long, and these values are placed in the input buffer of N. The output sampling rate of N is S1.

9.1.3 AudioSource and Sound2D

9.1.3.1 BIFS fields Characteristic

The pitch and speed change factors are restricted, if the url points to an HVXC object descriptor type.

- speed change factor: A possible variation is from 0.5 to 2.0 (defined as **spd** in ISO/IEC 14496-3, subpart 2, 5.5).
- pitch change factor: A possible variation is from 0.5 to 2.0 (defined as **pch_mod** in ISO/IEC 14496-3, subpart 2, 5.3).

9.1.3.2 Procedure to Test AudioSource Node

Testing the AudioSource+Sound2D Scene shall be performed:

by comparing the output of a decoder under test with a reference output supplied by the electronic attachment to this part of ISO/IEC 14496 using the procedure RMS measurement (7.1.2.2.1). This test only verifies the computational accuracy of an implementation (Test scenes AB001 to AB004);

by comparing the output of a decoder with the output of the same decoder in different instants of time along the sequence. To be called an ISO/IEC 14496-1 audio systems decoder, the decoder shall produce an output that changes in time according to position changes described in the scene. In test scenes AB005 to AB006 measurable changes shall be produced in the output of the decoder every 0.5 seconds, the time interval among position changes in the scene. This test verifies the spatial capabilities of the decoder.

9.1.3.3 Audio BIFS Test Scenes

AB001 One AudioSource node connected to one Sound2D node with default fields, except spatialize = FALSE, using CU1_Yx as input.

AB002, AB003, AB004 The same as AB001, with CU2_Yx, CU3_Yx, CU4_Yx as inputs, respectively.

AB005 One AudioSource node connected to one Sound2D node with default fields, except location, using CU1_Yx as input. The sound position describes a line in front of the listener, moving from -45° to 45° in azimuth. The location field is updated every 0.5 seconds and the source is moved by 15° from left to right at each update. The test stops 0.5 second after the 45° position has been reached.

AB006 The same as AB005 using CU4_Yx as input.

Template to code BIFS scenes:

```
Sound2D{
        AudioSource{
                url             2
                pitch           1
                speed           1
                NumChan         1
                PhaseGroup      [0]
        }
        intensity               1.0
        location                0,0
        spatialize              FALSE
}
```

Figure 15 — AB001: Sound2D has AudioSource as input.
Object descriptor with id 2 is referred to as the input audio stream (e.g. CU1).

For sequences AB001 to AB006 the electronic attachment to this part of ISO/IEC 14496 provides both a normative MP4 file and a textual parametric source like in the template of Figure 15, to be encoded by the decoder provider using the specific input CU and either the reference encoder or an equivalent. For sequences AB001 to AB004 the electronic attachment to this part of ISO/IEC 14496 provides reference output.

Table 83 — AudioBIFS test sequences

File Name	AB001	AB002	AB003	AB004	AB005	AB006
Content	BIFS	BIFS	BIFS	BIFS	BIFS	BIFS
Bitstream from a source (url)	CU1	CU2	CU3	CU4	CU1	CU4

9.1.4 AudioSource and Sound

9.1.4.1 BIFS fields Characteristic

The pitch and speed change factors are restricted, if the url points to an HVXC object descriptor type.

- speed change factor: A possible variation is from 0.5 to 2.0 (defined as **spd** in ISO/IEC 14496-3, subpart 2, 5.5).
- pitch change factor: A possible variation is from 0.5 to 2.0 (defined as **pch_mod** in ISO/IEC 14496-3, subpart 2, 5.3).

9.1.4.2 Procedure to Test AudioSource Node

Testing the AudioSource+Sound Scene shall be performed:

by comparing the output of a decoder under test with a reference output supplied by the electronic attachment to this part of ISO/IEC 14496 using the procedure RMS measurement (7.1.2.2.2). This test only verifies the computational accuracy of an implementation (Test scenes AB011 to AB014);

by comparing the output of a decoder with the output of the same decoder in different instants of time along the sequence. To be called an ISO/IEC 14496-1 audio systems decoder, the decoder shall produce an output that changes in time according to position changes described in the scene. In test scenes AB015 to AB016 measurable changes shall be produced in the output of the decoder every 0.5 seconds, the time interval among position changes in the scene. This test verifies the spatial capabilities of the decoder.

9.1.4.3 Audio BIFS Test Scenes

AB011 One AudioSource node connected to one Sound node with default fields, except spatialize = FALSE, using CU1_Yx as input.

AB012, AB013, AB014 The same as AB011, with CU2_Yx, CU3_Yx, CU4_Yx as inputs, respectively.

AB015 One AudioSource node connected to one Sound node with default fields, except location, using CU1_Yx as input. The sound position describes an arch at a distance of 2 meters from the listener, moving from -60° to 60° in azimuth. Heigth is constant at 2 meters for both, the Sound location and ListeningPoint. The location field is updated every 0.5 seconds and the source is moved by 15° clockwise at each update.

AB016 The same as AB005 using CU4_Yx as input.

For sequences AB011 to AB016 the electronic attachment to this part of ISO/IEC 14496 provides both a normative MP4 file and a textual parametric source, to be encoded by the decoder provider using the specific input CU and either the reference encoder or an equivalent. For sequences AB011 to AB014 the electronic attachment to this part of ISO/IEC 14496 provides reference output.

Table 84 — AudioBIFS test sequences

File Name	AB011	AB012	AB013	AB014	AB015	AB016
Content	BIFS	BIFS	BIFS	BIFS	BIFS	BIFS
Bitstream from a source (url)	CU1	CU2	CU3	CU4	CU1	CU4

9.1.5 AudioClip

9.1.5.1 BIFS fields Characteristic

None.

9.1.5.2 Procedure to Test AudioClip Node

Testing the AudioClip node shall be performed by comparing the output of a decoder under test with a reference output supplied by the electronic attachment to this part of ISO/IEC 14496 using the procedure RMS measurement (7.1.2.2.2). (Test scenes AB021 to AB024).

9.1.5.3 Audio BIFS Test Scenes

AB021 One AudioClip node, using CU1_Yx as input, connected to one Sound2D node with default fields, except spatialize = FALSE.

AB022 The same as AB021, with CU2_Yx as input.

AB023 One AudioClip node, using CU3_Yx as input, connected to one Sound node with default fields, except spatialize = FALSE.

AB024 The same as AB023, with CU4_Yx as input.

For sequences AB021 to AB024 the electronic attachment to this part of ISO/IEC 14496 provides a normative MP4 file, a reference output and a textual parametric source, to be encoded by the decoder provider using the specific input CU and either the reference encoder or an equivalent.

Table 85 — AudioBIFS test sequences

File Name	AB021	AB022	AB023	AB024
Content	BIFS	BIFS	BIFS	BIFS
Sequence from a source (url)	CU1	CU2	CU3	CU4

9.1.6 AudioSwitch

See also 9.1.2.2. Conformance Test of the click free capability of the AudioSwitch node is not required for decoders at Level 1 (sequence AB031).

9.1.6.1 BIFS fields Characteristic

None.

9.1.6.2 Procedure to Test Audio Node

Testing the AudioSwitch Scene for click free capability shall be performed by calculating the absolute value of the DFT of the output sequence AB031 (second 7 to 8) described later. It is defined as the pass band of the signal the frequency interval between 400Hz and 1kHz. The full length DFT of the output samples is calculated and its absolute value is taken in the interval from 0-sampling_rate/2, and the values are rescaled so that the peak component is 1. To be called an ISO/IEC 14496-1 audio systems decoder, the decoder shall provide an output such that the described absolute value is not greater than -20 dB in the two frequency intervals from 1-1.05 kHz and 380-400 Hz (5% of the pass band extremities) and not greater than -40 dB outside these two transition bands.

Testing the AudioSwitch node shall be performed by comparing the output of a decoder under test with a reference output supplied by the electronic attachment to this part of ISO/IEC 14496 using the procedure RMS measurement (7.1.2.2.2). (Test scenes AB032 to AB033).

9.1.6.3 Audio BIFS Test Scenes

AB031 Two AudioSource nodes connected to one AudioSwitch node with default fields and to a Sound2D with default fields (except spatialize at FALSE) using as inputs CU1 directly and CU1 followed by an AudioDelay node inserting a delay of 7 seconds. Switching is performed at a rate of 40 Hz, for 1 second, from second 7 to second 8 in performance time. The resulting output has a number of samples corresponding to the sampling rate.

AB032 Two AudioSource nodes connected to one AudioSwitch node with default fields, whichChoice initially set to [1 0], and to a Sound2D with default fields (except spatialize at FALSE) using as inputs CU1_Yx and CU2_Yx. Switching whichChoice to [0 1] is performed after 5.5 seconds.

AB033 The same as AB032, with CU3_Yx and CU4_Yx as inputs.

For sequence AB031 the electronic attachment to this part of ISO/IEC 14496 provides both a normative MP4 file and a textual parametric source, to be encoded by the decoder provider using the specific input CU and either the reference encoder or an equivalent.

For sequences AB032 and AB033 the electronic attachment to this part of ISO/IEC 14496 provides a normative MP4 file, a reference output and a textual parametric source, to be encoded by the decoder provider using the specific input CU and either the reference encoder or an equivalent.

Table 86 — AudioBIFS test sequences

File Name	AB031	AB032	AB033
Content	BIFS	BIFS	BIFS
Sequence 1 from a source (url)	CU1	CU1	CU3
Sequence 2 from a source (url)	CU1	CU2	CU4

9.1.7 AudioMix and Sampling Rate Conversion

See also 9.1.2.2. Testing of the Sampling Rate conversion is not required for Level 1 decoders. Testing of non integer ratios for Sampling Rate conversion is not required for Level 1 and Level 2 decoders.

9.1.7.1 BIFS fields Characteristic

None.

9.1.7.2 Procedure to Test AudioMix Node and SR conversion

Testing the AudioMix and SR conversion scene shall be performed by comparing the output of a decoder under test with a reference output supplied by the electronic attachment to this part of ISO/IEC 14496 using the procedure described in 9.1.2.3 “Sampling Rate conversion”. Software is provided for performing this verification procedure. To be called an ISO/IEC 14496-1 audio systems decoder, the decoder shall provide an output such that the SNR level of the difference signal between the output of the decoder under test and the supplied reference output quality shall be close in accuracy to the DAC that the signal is targeted for, i.e. according to the rule dB SNR = 6 * (nbits -1), where nbits is the number of bits corresponding to the maximum bit depth of any of the signals being so converted and/or composited. Close in accuracy means that this value shall be guaranteed at least for integer ratios, and could be slightly less for non-integer ratios (like 16000 to 22050).

Sequences to be used for test are AB041 to AB044 described later.

9.1.7.3 Audio BIFS Test Scenes

AB041 Two AudioSource nodes connected to one AudioMix node with default fields and to a Sound2D with default fields using CU2_Ya (8 kHz) and CU2_Yb (16 kHz) as inputs. Output is expected at 16 kHz. Levels are set to 1 and 0 respectively, i.e. only the 8 kHz source is audible. Performance stops after 5 seconds.

AB042 Three AudioSource nodes connected to one AudioMix node with default fields and to a Sound2D with default fields using CU2_Ya (8 kHz), CU2_Yb (16 kHz) and CU2_Yc (22.05 kHz) as inputs. Output is expected at 22.05 kHz. Levels are set to 0.5 for the first two channels, and to 0 for the third. It is not allowed to one of the two channels to terminate before the other, i.e. the two channels shall be synchronized on a sample per sample basis. Performance stops after 5 seconds.

AB043, AB044 The same as AB041, AB042 with CU3 as input.

For sequences AB041 to AB044 the electronic attachment to this part of ISO/IEC 14496 provides both a normative MP4 file and a textual parametric source, to be encoded by the decoder provider using the specific input CU and either the reference encoder or an equivalent. For sequences AB041 to AB044 the electronic attachment to this part of ISO/IEC 14496 also provides reference output.

Table 87 — AudioBIFS test sequences

File Name	AB041	AB042	AB043	AB044
Content	BIFS	BIFS	BIFS	BIFS
Bitstream 1 from a source (url)	CU2	CU2	CU3	CU3
Bitstream 2 from a source (url)	CU2	CU2	CU3	CU3
Bitstream 3 from a source (url)	-	CU2	-	CU3
Accuracy of mixing among groups (phaseGroup)	0	0	0	0

9.1.8 AudioDelay and AudioMix

9.1.8.1 BIFS fields Characteristic

None.

9.1.8.2 Procedure to Test AudioDelay + AudioMix scenes

Testing the AudioDelay node shall be performed by comparing the output of a decoder under test with a reference output supplied by the electronic attachment to this part of ISO/IEC 14496 using the procedure RMS measurement (7.1.2.2.2). (Test scenes AB051 and AB052).

9.1.8.3 Audio BIFS Test Scenes

AB051 One AudioSource node using CU1_Yx as input connected to an AudioDelay node with delay field set to 1 second, feeding one AudioMix node with matrix elements equal to 0.5; the AudioMix node receives a second AudioSource, using CU4_Yx as input, and it feeds a Sound node with default fields except spatialize = FALSE. The sources are stopped after 6 seconds.

AB052 The same as AB051, with the delay field of the AudioDelay node set to 1.5 seconds.

For sequences AB051 and AB052 the electronic attachment to this part of ISO/IEC 14496 provides a normative MP4 file, a reference output and a textual parametric source, to be encoded by the decoder provider using the specific input CU and either the reference encoder or an equivalent.

Table 88 — AudioBIFS test sequences

File Name	AB051	AB052
Content	BIFS	BIFS
Sequence from a source (url)	CU1 CU4	CU1 CU4

9.1.9 AudioBuffer

9.1.9.1 BIFS fields Characteristic

None.

9.1.9.2 Procedure to Test AudioBuffer Node

The AudioBuffer node, with pitch changes, implies an interpolation over the bufferized samples, with a normally non integer pitch factor. Test procedure is in this case similar to non integer Sampling Rate conversion. The decoder shall provide an output such that the SNR level of the difference signal between the output of the decoder under test and the supplied reference output quality shall be as close as possible in accuracy to the rule dB SNR = 6 * (nbits -1), where nbits is the number of bits corresponding to the maximum bit depth of any of the signals being so bufferized and/or composited. Close in accuracy means that this value

shall be guaranteed at least for integer pitch factors, and could be reasonably less for non-integer pitch factors. (Test scenes AB061 and AB062).

9.1.9.3 Audio BIFS Test Scenes

AB061 One AudioSource node, using CU1_Yx as input, connected to an AudioBuffer node with startTime set to 5 seconds, length set to 12 seconds and loop set to FALSE, feeding one Sound node with default fields, except spatialize = FALSE. The AudioBuffer pitch is changed to 0.5 after 11 seconds.

AB062 The same as AB061, but the pitch field of the AudioBuffer node is changed to 2 after 11 seconds.

For sequences AB061 and AB062 the electronic attachment to this part of ISO/IEC 14496 provides a normative MP4 file, a reference output and a textual parametric source, to be encoded by the decoder provider using the specific input CU and either the reference encoder or an equivalent.

Table 89 — AudioBIFS test sequences

File Name	AB061	AB062
Content	BIFS	BIFS
Sequence from a source (url)	CU1	CU1

9.1.10 AudioFX

See also 7.15

9.1.10.1 BIFS fields Characteristic

Restrictions on field values.

9.1.10.2 Procedure to Test AudioFX Node

The decoder is tested on functionality by comparing its output with a reference output supplied by the electronic attachment to this part of ISO/IEC 14496 using the procedure RMS measurement (7.1.2.2.2). This test only verifies the computational accuracy of an implementation.

9.1.10.3 Audio BIFS Test Scenes

AB101 One AudioSource node, using CU1, connected to one AudioFX node with Stripe orchestra and Score and to a Sound2D node with default fields, except spatialize = FALSE.

AB102 One AudioSource node, using CU4, connected to one AudioFX node with Stripe orchestra and Score and to a Sound2D node with default fields, except spatialize = FALSE.

AB103 One AudioSource node, using CU1, connected to one AudioFX node with AllComb orchestra and score and to a Sound2D node with default fields, except spatialize = FALSE.

AB104 One AudioSource node, using CU4, connected to one AudioFX node with AllComb orchestra and score and to a Sound2D node with default fields, except spatialize = FALSE.

For sequences AB101 to AB104 the electronic attachment to this part of ISO/IEC 14496 provides the Orchestras and Scores for the AudioFX nodes in textual format, a normative MP4 file and a textual parametric source, to be encoded by the decoder provider using the specific input CU and either the reference encoder or an equivalent. The electronic attachment to this part of ISO/IEC 14496 also provides reference output.

Table 90 — AudioBIFS test sequences

File Name	AB101	AB102	AB103	AB104
Content	Stripe	Stripe	AllComb	AllComb
Orchestra definition (orch)	Stripe	Stripe	AllComb	AllComb
Score definition (score)	Stripe	Stripe	AllComb	AllComb
Sequence 1 from a source (url)	CU1	CU4	CU1	CU4

9.2 Advanced Audio BIFS nodes

9.2.1 Introduction

Advanced AudioBIFS nodes are used for adding advanced 3-D audio processing functionalities for Virtual Reality rendering purposes in 3-D scenes.

9.2.2 Composition Unit Inputs

The input audio streams used in the conformance testing of Advanced AudioBIFS shall be outputs of a Null Object Type decoder, and they are monophonic sounds at 8 kHz or 22050 kHz sampling rate. They are explained below:

CU1_AAB_Px: Composition Unit Input PCM: Speech recorded in an anechoic chamber. Duration approximately 50 s. Sampling rate 8000 Hz

CU2_AAB_Px: Composition Unit Input PCM: An impulse sound with a value 1.0 of the first sample, followed by zeros. Duration approximately 20 s (160 000 samples). Sampling rate 8000 Hz

CU3_AAB_Px: Composition Unit Input PCM: Speech recorded in an anechoic chamber. Duration approximately 50 s. Sampling rate 22050 Hz

CU4_AAB_Px: Composition Unit Input PCM: An impulse sound with a value 1.0 of the first sample, followed by zeros. Duration approximately 20 s (441 000 samples). Sampling rate 22050 Hz

9.2.3 Compositor Output

The output of the audio compositor will be investigated for conformance, and shall be a single output, N channel (depending on the spatialization and reproduction method used) PCM audio stream. The input audio streams are at 16 bit signed integer sample format, and the processing defined by the Advanced Audio BIFS nodes in the scene will be carried out at an accuracy of at least 16 bits.

Because of the non-normative nature of implementing many of the Advanced AudioBIFS features (e.g., late reverberation), no sample-wise comparison is done to the output sound from the compositor. Some of the features can be evaluated in a static situation (no dynamic changes, such as sound source or viewpoint movements, in the 3-D environment) by measuring the *impulse response* of the compositor. Some functionalities, on the other hand, require testing in a dynamic situation where only subjective evaluation can be used (the user is listening to the sound compositor output, and watching the visual compositor output if visual components are present).

9.2.4 Physical Approach

This Clause describes the conformance testing for the rendering (audio output) of Advanced AudioBIFS nodes (physical approach, as described in 7.1.1.2.13.4 of ISO/IEC 14496-11:2005.

The BIFS nodes involved in the Advanced AudioBIFS (physical approach) are:

DirectiveSound, a node that is used as a topmost node of an AudioBIFS sub graph for attaching audio to 3-D scenes. It may contain an AudioBIFS sub graph similarly as Sound or Sound2D nodes, allowing for example

mixing of decoded audio streams that are outputs of different audio decoders, to a single sound track, thereby associating them with one physical source of sound in a 3-D scene.

AcousticScene that is a node that is used for defining rectangular 3-D regions within which source sound of DirectiveSound node is heard. It is also used for binding together reflective or sound obstructing surfaces that are involved in the same auralization process (processing and rendering of sound according to physical description of an acoustic environment), and adding common late reverberation characteristics to sounds that are positioned inside the defined region.

AcousticMaterial, a node that is used for attaching visual and acoustic properties to IndexedFaceSet nodes that act as sound reflecting and obstructing surfaces that are bound together by defining an AcousticScene. In order to be involved in an auralization process these IndexedFaceSets have to be defined in Geometry nodes that are siblings or exist in the sibling sub graphs of an AcousticScene node.

Some functionalities of the Advanced AudioBIFS can be objectively tested (i.e., measured from an impulse response of a digital filter (DSP) structure used in the advanced audio rendering process), whereas some of the features can be verified only perceptually (by listening to the sound output of the system).

In the following, the BIFS components needed for the conformance testing are listed and then the methods for testing each functionality are explained.

Scenes are provided in a textual format (textual BIFS scene graphs) with the conformance bit streams as mp4 files. The textual format scenes provide a detailed documentation of what should be the compositor output (the decoded scene, including the perceived sound output or recorded impulse response characteristics), and the corresponding .mp4 compressed data files should produce the described scenes when they are composited with the MPEG-4 decoder that is being tested.

9.2.4.1 BIFS components needed in the conformance testing

Advanced AudioBIFS nodes are used for advanced modeling of sound sources and sound propagation in 3-D Virtual Reality applications. These applications can be audio only (for creating time varying 3-D room acoustic effects, for example), or audiovisual applications where the Advanced AudioBIFS nodes can be used for creating dynamic and synchronized modeling of sound propagation from the source to the listening point (defined by a **Viewpoint** or a **ListeningPoint** node) which aims at enhanced and immersive perception of an audiovisual 3-D space. In the conformance testing of the physical approach of the Advanced AudioBIFS, each scene includes a minimal set of nodes and behaviour that is needed to test a certain functionality of the node or its field.

To test the conformance of all the functionalities of the Advanced AudioBIFS nodes, the following BIFS nodes in addition to the Advanced AudioBIFS nodes are needed:

Root node that is used as a top-most node in all the BIFS scenes for binding together all the scene information in one BIFS session.

One of the 3-D grouping nodes (Transform, Group, OrderedGroup, ...) for binding an AcousticScene node to a set of acoustically responding surfaces (IndexedFaceSet nodes) when that functionality is needed.

Viewpoint or ListeningPoint node that is used for defining the listening point according to which the spatial properties of sound are computed.

IndexedFaceSet, Geometry, and Appearance that form a visual polygonal surface to which acoustic (and visual) properties can be attached with an AcousticMaterial node (node in the material field of Appearance).

AudioSource with a url pointing to an elementary audio stream. This node is used as the only AudioBIFS node in the source field of the DirectiveSound node, and the sound it is pointing to is single-channel audio in conformance testing of Advanced AudioBIFS nodes. This is done due to the fact that the main purpose of these nodes is to add advanced features to the spatial processing of sound. And in the case of multichannel input sound, if the phaseGroup flag of any of the input streams is set to TRUE, no spatialization is done, and if it is set to FALSE, the input channels of DirectiveSound are first summed to form a single monophonic channel before any spatialization is carried out.

To perform tests for scenes in dynamic conditions (where either the **DirectiveSound**, the listening point (**Viewpoint** or **ListeningPoint** node), or one or more of the polygons having acoustical relevance are moving (**IndexedFaceSets** with **AcousticMaterial**, as explained in ISO/IEC 14496-1). In these tests the movement is achieved by animating one of these components; The **Viewpoint** can typically be animated by user input (e.g., navigation with an input device such as mouse of a computer). However, in the conformance tests, dynamic situations are caused by routing **TimeSensor** events to **PositionInterpolator** or **OrientationInterpolator** which are again used to change values of the translation and rotation fields of a **Transform** node that is a parent node of the animated objects. These additional scene components are thus:

TimeSensor (See 7.2.2.129 of ISO/IEC 14496-11:2005)

PositionInterpolator (See 7.2.2.102 of ISO/IEC 14496-11:2005) and/or OrientationInterpolator (7.2.2.91, ISO/IEC 14496-11:2005)

ROUTE syntax (See 7.1.1.2.8.1.3 of ISO/IEC 14496-11:2005 is used to route the values of PositionInterpolator and OrientationInterpolator to the field values of the Transform node according to the time fractions of TimeSensor.

Additionally, to give visual body to sound sources (for objective testing and audiovisual interaction), the following geometry nodes are used in the test scenes:

Sphere

Box

Circle

And to give visual appearance to the geometry objects, **Appearance** and **Material** nodes are associated to these objects. When a visual sound source is formed, **DirectiveSound** is bound to a geometry node (or a grouping node composed of several **Geometry** nodes) with a **Transform** node that can be used to group the **DirectiveSound** node and the associated visual object together and to place them in an arbitrary (and time-varying) position in a 3-D scene.

9.2.4.2 Conformance testing procedure

Testing all the functionalities of the **DirectiveSound** requires the set of BIFS components listed in the previous Subclause. Nevertheless, testing of some subsets of its functionalities does not require all those components. For example, if the spatialization, distance dependent attenuation, or air absorption is tested, no **AcousticMaterial** (and **IndexedFaceSets**), or animated dynamic movement (requiring **TimeSensor**, and **Position**- and/or **OrientationInterpolator** + ROUTEs) are needed.

In the following, the conformance testing of the physical approach of Advanced AudioBIFS is divided into separate testing of **DirectiveSound**, **AcousticScene**, and **AcousticMaterial**. For each of these nodes separate tests are also carried out for testing all of their functionalities (i.e., those that are enabled by the different fields of these nodes).

The testing of these nodes is divided to two categories. One is referred to as *objective testing,* meaning using impulse sound as an input signal that the **AudioSource** url points to, and calculating and comparing properties of the response of the Advanced Audio compositor (by recording the output of the compositor digitally) to the values given in the fields of the Advanced AudioBIFS nodes. This testing method is can only be done in a static situation where the response to an excitation signal can be considered that of a LTI (linear time invariant) system. The other test method gives *subjective* results (verified by listening to the compositor output), and it can also be done in time-varying (dynamic) conditions where one of the scene components move, thus causing a time-variant effect (e.g., in testing the Doppler effect, or a situation where the acoustic conditions change, for example when moving from one room to another). All the different scene setups (static and dynamic) are tested with the latter method, and a part of them (the static ones) with the former one, i.e., by measuring and evaluating the impulse response. All the setups are also tested with two different sampling rates.

The tests are categorized into testing of **DirectiveSound**, **AcousticScene**, and **AcousticMaterial**. **DirectiveSound** node is needed in all the tests to reveal the functionalities of the other two nodes. In the physical approach, the testing of **DirectiveSound** and **AcousticMaterial** always require also the presence of **AcousticScene**, and **AcousticMaterial** has no effect without the presence of **DirectiveSound** and **AcousticScene**. In 9.2.4.3, 9.2.4.4, and 9.2.4.5 the testing procedures are described for **DirectiveSound**, **AcousticScene**, and **AcousticMaterial,** respectively, so that all the functionalities of these three nodes are covered.

9.2.4.3 Procedure to test DirectiveSound

Below is **DirectiveSound** node and its fields listed with their default values:

DirectiveSound {

angles	0
directivity	1
frequency	[]
speedOfSound	340
distance	100
useAirabs	FALSE
direction	0, 0, 1
intensity	1
location	0, 0, 0
source	NULL
perceptualParameters	NULL
roomEffect	FALSE
spatialize	TRUE

}

This Subclause describes the testing of the following fields:

angles

directivity

frequency

speedOfSound

distance

spatialize

useAirabs

direction

In testing these fields, the fields intensity, roomEffect, and perceptualParameters shall be set to their default values.

The following nodes are involved in the testing of **DirectiveSound:**

Advanced AudioBIFS nodes:

DirectiveSound

AcousticScene

Other nodes used in the scenes:

Root

Viewpoint

Transform

TimeSensor

OrientationInterpolator

Geometry

Geometry nodes: Box, IndexedFaceSet, Circle

Appearance

9.2.4.3.1 Testing of directivity of a sound source.

Of all fields of the **DirectiveSound** node, angles, directivity, frequency, and direction fields are used to define the directivity of a 3-D source, i.e., the non-uniform radiation pattern to different directions with respect to the vector defined by the direction field of this sound.

9.2.4.3.1.1 Scene configuration and field characteristics of DirectiveSound and AcousticScene

In these tests, **DirectiveSound** fields useAirabs and spatialize fields are set to FALSE, and distance and speedOfSound fields are set to 0. The direction field is set to 0 0 1 (pointing to the direction of a positive z-axis, towards the **Viewpoint**). The direction of the **DirectiveSound** source is changed with **Transform** node.

AcousticScene is included in the scene with the default values (infinite audibility region with no reverberation).

For testing this property, the directivity of the source is defined by the directivity and angles fields, and in some of them the frequency field. In all the tests, the number of angles (length of the angles field) is 3. Objective testing is done by measuring and evaluating the impulse response of the Compositor output at each of the angles defined in the angles field. Subjective testing is carried out by rotating the DirectiveSound (and the associated visual sound source object) node with a help of a **Transform**, **TimeSensor**, and **OrientationInterpolator** nodes.

A visual object composed of **Geometry** nodes is included in the scene to give a physical body to the sound source. The **Transform** node groups together the visual object and the **DirectiveSound** node.

9.2.4.3.1.2 Test Scenes

Scenes for objective testing:

AABphy1-3 These scenes are used for testing of frequency independent directivity. The impulse response of the system shall be measured at all three defined angles of directivity (one measurement corresponding to one of these scenes). For each angle, it the response should only include the delay of one update interval corresponding to the update rate of the audio scene parameters, with respect to the direct sound, and after the delay the gain of the output should be the same as the gain of the directivity field value for the angle being tested. The input sound for this scene is CU2_AAB_Px.

AABphy4-6 These scenes are used for testing of frequency dependent directivity expressed in filter coefficient form. The impulse response of the system shall be measured at all three defined angles of directivity (one measurement corresponding to one of these scenes). For each angle, it should include only the delay of one update interval corresponding to the update rate of the audio scene parameters, with respect to the direct sound, after which the output should be that defined by the filter coefficient for that angle. The input sound for this scene is CU2_AAB_Px.

AABphy7-9 These scenes are used for testing of frequency dependent directivity expressed as gain-frequency pairs. The impulse response of the system is measured at all three angles (one measurement corresponding to one of these scenes), and the given frequency magnitude response should be matched with an accuracy of 1 dB at the frequencies specified by the frequency field. The magnitude response is computed from the measured impulse response after the delay introduced by the update interval of the audio parameters. The input sound for this scene is CU2_AAB_Px.

AABphy10-18 Same as AABphy1-9, respectively, but with CU4_AAB_Px as input signal.

Scenes for subjective testing:

AABphy19 This scene is used for subjective evaluation of frequency independent directivity. Audiovisual source is rotated, and the changes in frequency independent directivity should be heard as smoothly changing, and it must not produce audible artifacts (transitions when changing from one directivity angle to another). CU1_AAB_Px.

AABphy20 This scene is used for subjective evaluation of frequency dependent directivity in filter coefficient form. Audiovisual source is rotated, and the changes in frequency dependent directivity should be heard as smoothly changing, and it must produce no audible artifacts (transitions when changing from one directivity angle to another). Input sequence is CU1_AAB_Px.

AABphy21 This scene is used for subjective evaluation of frequency dependent directivity expressed as gain-frequency pairs. Audiovisual source is rotated, and the changes in frequency dependent directivity should be heard smoothly changing but produce no audible artifacts (transitions when changing from one directivity angle to another). CU1_AAB_Px.

AABphy22-24 Same as AABphy19-21, respectively, but with CU3_AAB_Px as input signal.

9.2.4.3.2 Testing of spatialize field

The spatialize field indicates whether the incident angle of the arriving sound is rendered. The method for spatializing sound is non-normative; therefore only subjective testing is performed.

9.2.4.3.2.1 **Scene configuration and field characteristics of** DirectiveSound **and** AcousticScene

In these tests, **DirectiveSound** field useAirabs is set to FALSE, spatialize field is set to TRUE, and the speedOfSound and distance fields are set to 0.

AcousticScene is included in the scene with the default values (infinite audibility region with no reverberation).

A visual **Sphere** node is associated to **DirectiveSound** node with a help of a **Transform** node. **TimeSensor** and **PositionInterpolator** nodes are used for moving the source dynamically.

In these scenes the sound source moves from –60 to 60 degrees in azimuth, and it should sound like the source is moving from left to right with respect to the listener orientation.

9.2.4.3.2.2 Test scenes

Scenes for subjective testing:

AABphy25 In this scene an audiovisual source is first positioned to –60 degrees for five seconds in azimuth angle with respect to the listener, after which it is moved to zero degrees, where it stays for 5 seconds, and then it is moved to +60 degrees. The movements from –60 to 0 degrees and from 0 to 60 degrees each last 5 seconds, and there should be no audible lag in the movement or stopping of sound with respect to the visual source movement (or to the description of movement, if visual parts are not implemented in a corresponding profile). The user should hear these effects with no audible artifacts (transitions in the spatialization processing of sound). The input sequence used in this test is CU1_AAB_Px.

AABphy26 The same scene and procedure of testing as above, but using CU3_AAB_Px as input sequence.

9.2.4.3.3 Testing of distance field

The value of the distance field defines the distance dependent attenuation of sound in a scene. Both objective and subjective tests are carried out for testing of this property, and the test is carried out for one value of the distance field. In the distance dependent attenuation the sound is attenuated linearly on a dB scale as a function of distance from one meter to the defined value of distance field. Thus, the gain applied to sound when the distance between the source and the listener locations is between one meter and the value of distance field in meters is given by:

$$g = 10^{-3\cdot(s-1)/(dist-1)},$$

where *s* is the current distance, and *dist* is the value of the distance field. When the distance between the **Viewpoint** and **DirectiveSound** is less than one meter, the gain *g* is one and beyond the distance *dist*, the sound is not audible.

9.2.4.3.3.1 Scene configuration and field characteristics of DirectiveSound and AcousticScene

In these tests, **DirectiveSound** field useAirabs and spatialize fields are set to FALSE, the speedOfSound is set to 0, and distance field is set to 100.

AcousticScene is included in the scene with the default values (infinite audibility region with no reverberation).

A visual **Sphere** node is associated to **DirectiveSound** node with a help of a **Transform** node. **TimeSensor** and **PositionInterpolator** nodes are used for moving the source dynamically in subjective tests. Objective tests measure the level of the impulse response at three different distances between the source and the listener.

9.2.4.3.3.2 Test scenes

Scenes for objective testing:

AABphy27-29 In scenes 27-29 the source is positioned at 1.0 m, 50 m, and 100 m distances from the **Viewpoint.** The response of the audio compositor is measured, and the initial delay before the first compositor output should be no more than the update interval of the audio scene parameters. The input sequence is CU2_AAB_Px, and the level of the compositor output signal is compared to the first sample of the audio input signal, and the level of the output should match the gain computed from the equation in 9.2.4.3.4.1, when *s* is 1.0, 50.0, and 100.0, respectively, with an accuracy of 1 dB or better.

AABphy30-32 Same as the above test, but the input sequence is CU4_AAB_Px.

Scenes for subjective testing:

AABphy33 In this test the source is first at 1.0 meter distance, where it stays for 5 seconds, then it is moved to 50.0 m distance during 10 seconds, and stays there for 5 seconds, and finally it is moved to 100.1 meter distance during 10 seconds. The listener should hear the sound attenuating smoothly during the source movements, and the level remaining the same during the stop at 50 meters, and being inaudible at 100.1 meters. CU1_AAB_Px.

AABphy34 Same as the above test, but with input signal CU3_AAB_Px.

9.2.4.3.4 Testing of speedOfSound field

Field speedOfSound defines the initial delay introduced to sound when the source and the viewpoint are in different locations in the scene. The delay simulates the propagation delay of sound in a medium and is dependent on the distance between the source and the listener, and the speed of sound propagation in the medium. This field is tested both objectively and subjectively. This test is carried out for two different values of speedOfSound, and by positioning the source at two different distances from the **Viewpoint**. The delay that should be applied to sound is given in seconds by:

$$d = \frac{s}{speedOfSound},$$

where *speedOfSound* is the value given by the field of a same name, and *s* is the distance between the **Viewpoint** and the **DirectiveSound** location.

9.2.4.3.4.1 Scene configuration and field characteristics of DirectiveSound and AcousticScene

In these tests, **DirectiveSound** fields useAirabs and spatialize are set to FALSE, the distance field is set to 0, and the speedOfSound field is set to 340 or 170 (depending on the test scene under consideration).

AcousticScene is included in the scene with the default values (infinite audibility region with no reverberation).

A visual **Sphere** node is associated to **DirectiveSound** node with a help of a **Transform** node. **TimeSensor** and **PositionInterpolator** nodes are used for moving the source dynamically in the subjective tests.

9.2.4.3.4.2 Test scenes

Objective testing of speedOfSound:

AABphy35-36 The speedOfSound is given a value 340, and in these two scenes the delay it causes is measured from the compositor output when replacing the sound source at 0 and 100 meter distance from the **Viewpoint**. The 0 distance must cause no delay to sound, and the delay at 100-meter distance should match that calculated from the equation in 0. with an accuracy of 10% or better. The delay is considered as the time lag (in addition to the lag introduced by the update interval of audio scene updates) in the response of the compositor to sound CU2_AAB_Px.

AABphy37-38 Same as the test above, but CU4_AAB_Px is used as an input sound to the compositor.

AABphy39-40 Same as AABphy35-36 but with a value 170 given to speedOfSound.

AABphy41-42 Same as above but for input sound CU4_AAB_Px.

Subjective testing of speedOfSound:

AABphy43-44 Value of speedOfSound is 340 and 170 in these scenes, respectively. The sound source is moved from 100 distance to 0 distance and back to 100 distance. First the sound is at 100-meter distance for 5 seconds. Then it moves to 0 distance and back to 100 distance (simulating a source passing the listener) with a uniform speed during 10 s. The changing delay should be heard as a Doppler effect causing a raise in the pitch of sound when the source is getting closer to the listener, and a decrease in the pitch of sound when it is drawn away from the listener. The changing delay has to be interpolated between samples so that the Doppler effect is heard and no artifacts such as clicks are audible. The change in the pitch should be heard twice as strong in AABphy44 than in AABphy43. The input test signal is CU1_AAB_Px.

AABphy45-46 The same scenes as for AABphy43-44 but with input sound CU3_AAB_Px.

9.2.4.3.5 Testing of useAirabs field

The useAirabs field is used to enable distance dependent lowpass filtering caused by the stronger sound absorption of air at high frequencies than at low frequencies. The testing of this field is done both objectively and subjectively.

9.2.4.3.5.1 Scene configuration and field characteristics of DirectiveSound and AcousticScene

In these tests, **DirectiveSound** field useAirabs is set to TRUE, and spatialize field is set to FALSE, the speedOfSound and the distance fields are set to 0.

AcousticScene is included in the scene with the default values (infinite audibility region with no reverberation).

A visual **Sphere** node is associated to **DirectiveSound** node with a help of a **Transform** node. **TimeSensor** and **PositionInterpolator** nodes are used for moving the source dynamically in subjective tests. Objective tests measure the level of the impulse response at three different distances between the source and the listener.

9.2.4.3.5.2 Test scenes

Objective testing of useAirabs:

AABphy47-49 An impulse response of the compositor output is measured and a magnitude response is computed from it at distances 10, 50, and 100 meters between the **DirectiveSound** source and the **Viewpoint** in scenes AABphy47-49. The magnitude response must match that computed from formula 5 in ISO 9613-1 at the given distance with an accuracy of 10% or better. The parameters concerning the atmospheric conditions are: humidity = 70%, temperature = 20 degrees centigrade, air pressure = 101325 Pa. The input sequence used in this test is CU2_AAB_Px.

AABphy50-52 Same as the above test but with CU4_AAB_Px as the input source signal.

Subjective testing of useAirabs:

AABphy53 In this test a **DirectiveSound** source is dynamically moving from a 1-meter distance to 100-meter distance. The air absorption filtering should be heard as increased lowpass filtering as a function of distance. No audible artifacts such as clicks should be heard as the filtering changes. The input sound used in this test is CU1_AAB_Px

AABphy54 Same as the above test but with CU3_AAB_Px.

9.2.4.4 Procedure to test AcousticScene

Below is the node interface for **AcousticScene** node

AcousticScene {

center	0 0 0
size	-1 –1 –1
reverbTime	0
reverbFreq	1000
reverbLevel	0.4
reverbDelay	0.5

}

This Subclause describes testing methods for testing all the fields of **AcousticScene.**

The following nodes are used in the testing of **AcousticScene:**

Advanced AudioBIFS nodes:

DirectiveSound

AoucsticScene

Other nodes used in these scenes:

Root

Viewpoint

Transform and Group

Geometry

IndexedFaceSet

Appearance

Material

9.2.4.4.1 Testing of late reverberation

Late reverberation properties are defined by the fields reverbTime, reverbFreq, reverbLevel, and reverbDelay.

9.2.4.4.1.1 Scene configuration and field characteristics of DirectiveSound and AcousticScene

The **DirectiveSound** node is included with the default values of fields except that the spatialize flag is set to FALSE for the objective testing of late reverberation, and the roomEffect is set to TRUE in all the tests.

9.2.4.4.1.2 Test scenes

Objective testing of late reverberation:

AABphy55 In this scene the reverbLevel = 0.4, reverbDelay = 0.05, reverbFreq = 1000, and reverbTime = 1.5. The impulse response of the compositor is recorded, and the delay and the level of the first reverberator output after the direct sound must be like indicated by the fields reverbDelay and reverbLevel, respectively. The reverberation time of the response is calculated at the octave band of the given frequency according to ISO 3382 using the integrated impulse method and must match the given reverbTime with an accuracy of 10% or better. The reverberation level is calculated by summing the squared magnitudes of all the samples in the late reverberation response, and taking the square root of the result. The resulting value should match the reverbLevel value with an accuracy of 1 dB or better. Input sound is CU2_AAB_Px.

AABphy56 In this scene the reverbLevel = 0.4, reverbDelay = 0.1, reverbFreq = [0 2000 4000], and reverbTime = [2.5 2.2 0.7]. The measurement is done like in AABphy55 (according to ISO 3382). The reverberation times measured at the frequencies given by reverbFreq should match the values given by reverbTime with an accuracy of 10% or better., Input sound is CU2_AAB_Px.

AABphy57 Same procedure and field values as for AABphy55, but with input sound CU4_AAB_Px.

AABphy58 Same field values as for AABphy56, except that the reverbFreq = [0 2000 10000]. Same procedure measuring and comparing the response as for AABphy55, but with input sound CU4_AAB_Px.

Subjective testing of late reverberation:

AABphy59 The late reverberation characterizing fields have the same values as for scene AABphy55. The compositor output is listened to and it should sound reverberant. The input sequence is CU1_AAB_Px.

AABphy60 The late reverberation characterizing fields have the same values as for scene AABphy56. The compositor output is listened and it should sound reverberating longer and with a larger delay (with respect to the direct sound) than AABphy59. The input sequence is CU1_AAB_Px.

AABphy61 Same as AABphy59 but with input sequence CU3_AAB_Px.

AABphy62 Same as AABphy60 but with reverbFreq = [0 2000 10000], and input sequence CU3_AAB_Px.

9.2.4.4.2 Testing of the 3-D rendering region

This test is only carried out subjectively, and it involves two **AcousticScene** nodes with one **DirectiveSound** source in the rendering region of each of them, and the **Viewpoint** movement from one **AcousticScene** region to another. In these tests, **DirectiveSound** field useAirabs, roomEffect, are set to FALSE, and spatialize field is set to TRUE, and the speedOfSound is set to 340, and the distance fields are set to 100.

9.2.4.4.2.1 **Scene configuration and field characteristics of** DirectiveSound **and** AcousticScene

The scenes involve two **AcousticScene** nodes and two **DirectiveSound** nodes. The 3-D rectangular regions of the **AcousticScenes** are limited in space by the size and center fields, so that the regions overlap partly. The reverberation characteristics are defined differently for each **AcousticScene,** so that in the first one there

is no reverberation added to sound, and in the second there is reverberation defined by reverberation field values reverbTime = 1.8, reverbDelay = 0.05.

9.2.4.4.2.2 Test scenes

Subjective testing of 3-D rendering region

AABphy63 In this scene there are two visual rooms in the rendering region of each **AcousticScene.** When the viewpoint is inside one room, one **DirectiveSound** source is heard, and when the **Viewpoint** moves to the other room, two sources are heard simultaneously for a while in the overlapping part of the **AcousticScene** regions, and finally in the rendering region of the second room, only the second **DirectiveSound** is heard. When the **Viewpoint** moves outside of both rendering regions, no sound is heard. The first **DirectiveSound** source is not reverberated, and the second **DirectiveSound** source is reverberated according to the late reverberation characteristics given the same as in AABphy60. Input sounds for both sources are CU1_AAB_Px.

AABphy64 Same as the above, but with input sound CU3_AAB_Px, and late reverberation values same as in scene AABphy62.

9.2.4.5 Procedure to test AcousticMaterial

Below is the node interface of **AcousticMaterial**

AcousticMaterial {

reffunc	0
transfunc	1
refFrequency	[]
transFrequency	[]
ambientIntensity	0.2
diffuseColor	0.8, 0.8, 0.8
emissiveColor	0, 0, 0
shininess	0.2
specularColor	0, 0, 0
transparency	0

}

This Subclause describes testing of reffunc, transfunc, refFrequency, and transFrequency.

The following nodes are used in the testing of **AcousticScene:**

Advanced AudioBIFS nodes:

DirectiveSound

AcousticScene

AcousticMaterial

Other nodes used in these scenes:

Root

Viewpoint

Transform and Group

Geometry

IndexedFaceSet

Appearance

In these tests the fields of the **DirectiveSound** are set to default values, except that the roomEffect is set to TRUE, and spatialize field is set to FALSE for the objective tests. The **AcousticScene** field values are the default values for all these scenes.

9.2.4.5.1 Testing of reflectivity

9.2.4.5.1.1 Scene configuration and field characteristics of DirectiveSound and AcousticScene

In these tests one reflective surface is included in a scene. It is positioned in x-z plane, and both the **DirectiveSound** and the **Viewpoint** are placed at a distance of 10 meters from the surface, with a distance of 5 meters between the **DirectiveSound** and the **Viewpoint.** The sound is reflected specularily off the surface, producing an image source whose distance from the **Viewpoint** is sqrt(425).

9.2.4.5.1.2 Test scenes

Objective testing of reflectivity:

AABphy65 This scene is used for testing of frequency independent reflectivity. The reflectivity fields of **AcousticMaterial** are: reffunc = 1, refFrequency = []. The impulse response of the compositor output is recorded, and the delays of the direct sound and the reflection should match to the delay values computed according to the equation in 0, setting the distance between the source and the listener to 5, and between the (reflected) image sound source and the listener to sqrt(425). The attenuation of the direct sound and the reflection should match to those computed by the equation in 9.2.4.3.3. Input sound signal used in this test is CU2_AAB_Px.

AABphy66 This scene is used for a frequency dependent reflectivity. The reflectivity is expressed in a filter coefficient form. The delay of the reflection should be the same as in AABphy65, and the impulse response of the reflection should match that of the filter defined in the reffunc field, scaled by a distance dependent attenuation identical to that of AABphy65. Input sound signal used in this test is CU2_AAB_Px.

AABphy67 The scene is used for a frequency dependent reflectivity. The reflectivity is expressed as gain – frequency pairs defining a magnitude response of a digital filter. The delay of the reflection should be like in AABphy65, and the magnitude response of the reflection should match to that defined by the reffunc and refFrequency fields with an accuracy of 1 dB, scaled by a distance dependent attenuation identical to that of AABphy65. Input sound signal used in this test is CU2_AAB_Px.

AABphy68 The scene and testing procedure is the same as in AABphy65, but the input signal is CU4_AAB_Px.

AABphy69 The scene and testing procedure is the same as in AABphy66, but the input signal is CU4_AAB_Px.

AABphy70 The scene and testing procedure is the same as in AABphy67, but the input signal is CU4_AAB_Px.

9.2.4.5.2 Testing of sound obstruction

9.2.4.5.2.1 **Scene configuration and field characteristics of** DirectiveSound **and** AcousticScene

This Subclause describes the objective testing of transmission of sound through a sound obstructing surface. The sound is positioned so that it is at a 10-meter distance from the **Viewpoint**, and a sound obstructing surface is in the path between them. The delay of sound should match that computed according to the equation in 0, and the attenuation should match that defined by the equation in 9.2.4.3.3.

9.2.4.5.2.2 Test scenes

Objective testing of sound obstruction:

AABphy71 This scene is used for testing of frequency independent obstruction of sound (attenuation caused by the surface). The sound should be attenuated with a gain that is a product of the distance dependent gain and that defined by the transfunc field. Input sound signal used in this test is CU2_AAB_Px.

AABphy72 This scene is used for testing of frequency dependent sound obstruction. The transmission filter is defined by the transfunc field (transFreq is set to []), and the impulse response at the compositor output should match the defined filter output, scaled by the same distance dependent gain as that of AABphy71. Input sound signal used in this test is CU2_AAB_Px.

AABphy73 This scene is used for testing of frequency dependent sound obstruction defined as gain – frequency pairs. The magnitude response of the compositor output should match that defined by the transfunc and transFrequency fields with an accuracy of 1 dB, scaled by the same distance dependent gain as that of AABphy71. Input sound signal used in this test is CU2_AAB_Px.

AABphy74 Same testing procedure as for AABphy71, but with input sequence CU4_AAB_Px.

AABphy75 Same testing procedure as for AABphy72, but with input sequence CU4_AAB_Px.

AABphy76 Same testing procedure as for AABphy73, but with input sequence CU4_AAB_Px.

9.2.4.5.3 Subjective testing of sound obstruction and reflectivity

This Subclause defines test scenes for subjective testing of the **AcousticMaterial**. In these tests the spatialize field of source is set to TRUE.

AABphy77 This scene contains the same initial setup for positioning the **DirectiveSound** source, a single **IndexedFaceSet** surface with **AcousticMaterial** associated with it, and **Viewpoint** node as in test scenes AABphy65-70. During the test the **DirectiveSound** source starts moving towards the edge of the **IndexedFaceSet** surface. The delay and gain, and the direction of arrival of the reflection should change according to current position of the

DirectiveSound node with respect to the surface and the viewpoint, and no clicks should be heard when the delay of the reflection changes. The reflection becomes inaudible when the image source caused by the reflecting surface becomes invisible to the **Viewpoint.** The **DirectiveSound** source moves to the other side of the surface (with respect to the **Viewpoint**), and the transmission filtering should be heard when the surface appears between the source and the listening point. Input sequence used in this scene is CU1_AAB_Px.

AABphy78 Same as AABphy77 but with input sound CU3_AAB_Px.

AABphy79 In this scene there are 7 sound reflecting and obstructing surfaces forming a simple room configuration. The **DirectiveSound** source is positioned inside the room, and the **Viewpoint** can move freely inside and outside of the room. Inside the room, the reflections should be heard giving a slightly reverberating effect, and outside of the room the sound is attenuated according to the transfunc and transFrequency definitions of the surfaces. Input sequence used in this scene is CU1_AAB_Px.

AABphy80 Same as AABphy79 but with input sound CU3_AAB_Px.

9.2.5 Perceptual Approach

This Subclause describes the conformance testing for the rendering (audio output) of Advanced AudioBIFS nodes (perceptual approach), as described in 7.1.1.2.13.4 of ISO/IEC 14496-11:2005. The perceptual approach shall be applied for all the **DirectiveSound** nodes that contain a **PerceptualParameters** node, i.e. for which the **PerceptualParameters** field is different from NULL.

The BIFS nodes involved in the Advanced AudioBIFS (perceptual approach) are:

DirectiveSound, a node that is used as a topmost node of an AudioBIFS sub graph for attaching audio to 3-D scenes. It may contain an AudioBIFS sub graph similarly as Sound or Sound2D nodes, allowing for example mixing of decoded audio streams that are outputs of different audio decoders, to a single sound track, thereby associating them with one physical source of sound in a 3-D scene.

PerceptualParameters, a node that is used for attaching perceptual properties to a directive sound source (DirectiveSound) in order to simulate virtual room effects that do not need to relate to the geometrical and/or visual BIFS scene.

Some functionalities of the Advanced AudioBIFS can be objectively tested (i.e., measured from an impulse response of a digital filter (DSP) structure used in the advanced audio rendering process), whereas some of the features can be verified only perceptually (by listening to the sound output of the system).

In the following, the BIFS components needed for the conformance testing are listed, and then the methods for testing each functionality are explained.

Scenes are provided in a textual format (textual BIFS scene graphs) with the conformance bit streams as mp4 files. The textual format scenes provide a detailed documentation of what should be the compositor output (the decoded scene, including the perceived sound output or recorded impulse response characteristics), and the corresponding .mp4 bitstream files should produce the described scenes when they are composed with the MPEG-4 decoder that is being tested.

9.2.5.1 BIFS components needed in the conformance testing

Advanced AudioBIFS nodes are used for advanced modeling of sound sources and sound propagation in virtual 3-D worlds and immersive music or soundtracks. These applications can be audio only (for creating time varying 3-D room acoustic effects, for example), or audiovisual applications where the Advanced AudioBIFS nodes can be used for creating dynamic and synchronized modeling of sound propagation from the source to the listening point (defined by a **Viewpoint** or a **ListeningPoint** node) which aims at enhanced

and immersive perception of an audiovisual 3-D space. In the conformance testing of the perceptual approach of the Advanced AudioBIFS, each scene includes a minimal set of nodes and behavior that is needed to test a certain functionality of the node or its field.

To test the conformance of all the functionalities of the Advanced AudioBIFS nodes that are used in the perceptual approach, the following BIFS nodes in addition to these Advanced AudioBIFS nodes are needed:

Root node that is used as a top-most node in all the BIFS scenes for binding together all the scene information in one BIFS session.

Viewpoint or ListeningPoint node that is used for defining the listening point according to which the spatial properties of sound are computed.

AudioSource with a url pointing to an elementary audio stream. This node is used as the only AudioBIFS node in the source field of the DirectiveSound node, and the sound it is pointing to is single-channel audio in conformance testing of Advanced AudioBIFS nodes. This is done due to the fact that the main purpose of these nodes is to add advanced features to the spatial processing of sound. And in the case of multichannel input sound, if the phaseGroup flag of any of the input streams is set to TRUE, no spatialization is done, and if it is set to FALSE, the input channels of DirectiveSound are first summed to form a single monophonic channel before any spatialization is carried out.

To perform tests for scenes in dynamic conditions (where either the **DirectiveSound**, the listening point (**Viewpoint** or **ListeningPoint** nodes) are moving), the movement is achieved by animating one of these components.The **Viewpoint** can typically be animated by user input (e.g., navigation with an input device such as mouse of a computer). However, in the conformance tests, dynamic situations are caused by routing **TimeSensor** events to **PositionInterpolator** or **OrientationInterpolator** which are again used to change values of the translation and rotation fields of a **Transform** node that is a parent node of the animated objects. These additional scene components are thus:

TimeSensor (See 7.2.2.129 of ISO/IEC 14496-11:2005)

PositionInterpolator (See 7.2.2.102 of ISO/IEC 14496-11:2005) and/or OrientationInterpolator (ISO/IEC 14496-1 sublause 9.4.2.66)

ROUTE syntax (See 7.1.1.2.8.1.3 of ISO/IEC 14496-11:2005 is used to route the values of PositionInterpolator and OrientationInterpolator to the field values of the Transform node according to the time fractions of TimeSensor.

- Additionally, to give visual body to sound sources (for objective testing and audiovisual interaction), the following geometry nodes are used in the test scenes:

Sphere

Cylinder

And to give visual appearance to the geometry objects, **Appearance** and **Material** nodes are associated to these objects. When a visual sound source is formed, **DirectiveSound** is bound to a geometry node (or a grouping node composed of several **Geometry** nodes) with a **Transform** node that can be used to group the **DirectiveSound** node and the associated visual object together and to place them in an arbitrary (and time-varying) position in a 3-D scene.

9.2.5.2 Conformance testing procedure

Testing all the functionalities of the **DirectiveSound** requires the set of BIFS components listed in the previous Subclause. Nevertheless, testing of some subsets of its functionalities does not require all those components. For example, if the spatialization, distance dependent attenuation, or air absorption is tested,) no animated dynamic movement (requiring **TimeSensor**, and **Position**- and/or **OrientationInterpolator** + ROUTEs) are needed.

In the following, the conformance testing of the perceptual approach of Advanced AudioBIFS is divided into separate testing of **DirectiveSound**, and **PerceptualParameters**. For each of these nodes separate tests are also carried out for testing all of their functionalities (i.e., those that are enabled by the different fields of these nodes).

The testing of these nodes is divided to two categories. One is referred to as *objective testing,* meaning using impulse sound as an input signal that the **AudioSource** url points to, and calculating and comparing properties of the response of the Advanced Audio compositor (by recording the output of the compositor digitally) to the values given in the fields of the Advanced AudioBIFS nodes. This testing method can only be done in a static situation where the response to an excitation signal can be considered that of a LTI (linear time invariant) system. The other test method gives *subjective* results (verified by listening to the compositor output), and it can also be done in time-varying (dynamic) conditions where one of the scene components move, thus causing a time-variant effect (e.g., in testing the Doppler effect, or a situation where the acoustic conditions change, for example when moving from one room to another). All the different scene setups (static and dynamic) are tested with the latter method, and a part of them (the static ones) with the former one, i.e., by measuring and evaluating the impulse response. All the setups are also tested with two different sampling rates.

The tests are categorized into testing of **DirectiveSound** and **PerceptualParameters**. In the perceptual approach, the testing of **DirectiveSound** always require also the presence of **PerceptualParameters**, and vice-versa. In 9.2.5.3 and 9.2.5.4 the testing procedures are described for **DirectiveSound**, and **PerceptualParameters** respectively.

9.2.5.3 Procedure to test DirectiveSound

Below is **DirectiveSound** node and its fields listed with their default values:

DirectiveSound {

angles	0
directivity	1
frequency	[]
speedOfSound	340
distance	100
useAirabs	FALSE
direction	0, 0, 1
intensity	1
location	0, 0, 0
source	NULL
perceptualParameters	NULL
roomEffect	FALSE
spatialize	TRUE

}

This Subclause describes the testing of the following fields:

angles

directivity

frequency

speedOfSound

distance

spatialize

useAirabs

direction

In testing these fields, the fields intensity and roomEffect shall be set to their default values. The perceptualParameters field shall contain a reference to a **PerceptualParameters** node for which the parameters fields are set to their default values unless otherwise stated.

The following nodes are involved in the testing of **DirectiveSound:**

Advanced AudioBIFS nodes:

- **DirectiveSound**
- **PerceptualParameters**

Other nodes used in the scenes:

- **Group**
- **Viewpoint**
- **DirectionalLight**
- **Transform**
- **Shape**
- **Appearence**
- **Material**
- **Sphere**
- **Cylinder**
- **TimeSensor**
- **OrientationInterpolator**

Note: the above mentioned fields are tested in the perceptual approach in a similar way as in the physical approach (i.e. that the rendering is identical in the two approaches).

9.2.5.3.1 Testing of directivity of a sound source.

Of all fields of the **DirectiveSound** node, angles, directivity, frequency, and direction are used to define the directivity of a 3-D source, i.e., the non-uniform radiation pattern to different directions with respect to the vector defined by the **direction** field of this sound.

9.2.5.3.1.1 Scene configuration and field characteristics of DirectiveSound

In these tests, **DirectiveSound** fields useAirabs and spatialize are set to FALSE, and distance and speedOfSound fields are set to 0. The direction field is set to 0 0 1 (pointing to the direction of a positive z-axis, towards the **Viewpoint**). The direction of the **DirectiveSound** source is changed with **Transform** node.

For testing this property, the directivity of the source is defined by the **directivity**, the **angles**, and the **frequency** fields. In all the tests, the number of angles (length of the angles field) is 3. Objective testing is done by measuring and evaluating the impulse response of the Compositor output at each of the angles defined in the angles field. Subjective testing is carried out by rotating the **DirectiveSound** (and the associated visual sound source object) node with a help of a **Transform**, **TimeSensor**, and **OrientationInterpolator** nodes.

A visual object composed of **Geometry** nodes is included in the scene to give a physical body to the sound source. The **Transform** node groups together the visual object and the **DirectiveSound** node.

9.2.5.3.1.2 Test Scenes

Scenes for objective testing:

AABper1-3 These scenes are used for testing of frequency independent directivity. In that case, only one frequency is defined in the **frequency** field. The impulse response of the system shall be measured at all three defined angles of directivity (one measurement corresponding to one of these scenes). For each angle, the response should only include the delay of one update interval corresponding to the update rate of the audio scene parameters, with respect to the direct sound, and after the delay the gain of the output should be the same as the gain of the directivity field value for the angle being tested. The input sound for this scene is CU2_AAB_Px.

AABper4-6 These scenes are used for testing of frequency dependent directivity expressed as gain-frequency pairs. The impulse response of the system is measured at all three angles (one measurement corresponding to one of these scenes), and the given frequency magnitude response should be matched with an accuracy of 1 dB at the frequencies specified by the frequency field. The magnitude response is computed from the measured impulse response after the delay introduced by the update interval of the audio parameters. The input sound for this scene is CU2_AAB_Px.

AABper7-12 Same as AABper1-6 but with CU4_AAB_Px as input signal.

Scenes for subjective testing

AABper13 This scene is used for subjective evaluation of frequency independent directivity. Audiovisual source is rotated, and the changes in frequency independent directivity should be heard as smoothly changing, and it must not produce audible artifacts (transitions when changing from one directivity angle to another). The input sound for this scene is CU1_AAB_Px.

AABper14 This scene is used for subjective evaluation of frequency dependent directivity expressed as gain-frequency pairs. Audiovisual source is rotated, and the changes in frequency dependent directivity should be heard smoothly changing but produce no audible artifacts (transitions when changing from one directivity angle to another). The input sound for this scene is CU1_AAB_Px.

AABper15-16 Same as AABper 13-14 but with CU3_AAB_Px as input signal.

9.2.5.3.2 Testing of spatialize field

The spatialize field indicates whether the incident angle of the arriving sound is rendered. The method for spatializing sound is non-normative; therefore only subjective testing is performed.

9.2.5.3.2.1 Scene configuration and field characteristics of DirectiveSound

In these tests, **DirectiveSound** field useAirabs is set to FALSE, spatialize field is set to TRUE, and the speedOfSound and distance fields are set to 0.

A visual object is associated to the **DirectiveSound** node with a help of a **Transform** node. **TimeSensor** and **PositionInterpolator** nodes are used for moving the source dynamically.

In these scenes the sound source moves from –60 to 60 degrees in azimuth, and it should sound like the source is moving from left to right with respect to the listener orientation.

9.2.5.3.2.2 Test scenes

Scenes for subjective testing:

AABper17	In this scene an audiovisual source is first positioned to –60 degrees for five seconds in azimuth angle with respect to the listener, after which it is moved to zero degrees, where it stays for 5 seconds, and then it is moved to +60 degrees. The movements from –60 to 0 degrees and from 0 to 60 degrees each last 5 seconds, and there should be no audible lag in the movement or stopping of sound with respect to the visual source movement (or to the description of movement, if visual parts are not implemented in a corresponding profile). The user should hear these effects with no audible artifacts (transitions in the spatialization processing of sound). The input sequence used in this test is CU1_AAB_Px.
AABper18	The same scene and procedure of testing as above, but using CU3_AAB_Px as input sequence.

9.2.5.3.3 Testing of distance field

The value of the distance field defines the distance dependent attenuation of sound in a scene *:* Within **distance** meters from the source, the sound is multiplied by the value of the **intensity** field before any spatial processing (directivity filtering, spatialization, or room effect). Outside this distance from the sound source, the sound is not audible. Between 0 and **distance**, the distance attenuation is performed according to paragraph 9.4.2.78.1.1 of ISO/IEC 14496-1 by modifying the source presence Es. If, however, the **distance** field is set to 0, no distance dependent attenuation is applied.

9.2.5.3.3.1 Scene configuration and field characteristics of DirectiveSound

In these tests, **DirectiveSound** fields useAirabs,spatialize and roomeffects are set to FALSE, the speedOfSound is set to 0, and distance field is set to 100. The directivity is uniform, both relative to the position of the source (angle) and to the frequency.

A visual object is associated to the **DirectiveSound** node with a help of a **Transform** node. **TimeSensor** and **PositionInterpolator** nodes are used for moving the source dynamically in subjective tests. Objective tests measure the level of the impulse response at three different distances between the source and the listener.

9.2.5.3.3.2 Test scenes

Scenes for objective testing:

AABper19-21	In scenes 19-21 the source is positioned at 1.0 m, 50 m, and 100 m distances from the **Viewpoint.** The response of the audio compositor is measured, and the initial delay before the first compositor output should be no more than the update interval of the audio scene parameters. The input sequence is CU2_AAB_Px, and the level of the compositor output signal is compared to the first sample of the audio input signal, and the level of the output should match the gain computed from the equation in 7.2.2.93.2.2 of ISO/IEC 14496-11:2005 when *s* is 1.0, 50.0, and 100.0, respectively with an accuracy of 1 dB or better.
AABper22-24	Same as the above test, but the input sequence is CU4_AAB_Px.

Scenes for subjective testing:

AABper25	In this test the source is first at 1.0 meter distance, where it stays for 5 seconds, then it is moved to 50.0 m distance during 10 seconds, and stays there for 5 seconds, and finally it is moved to 100.1 meter distance during 10 seconds. The listener should hear the sound attenuating smoothly during the source movements, and the level remaining the same during the stop at 50 meters, and being inaudible at 100.1 meters. The input sequence is CU1_AAB_Px.
AABper26	Same as the above test, but with input signal CU3_AAB_Px.

9.2.5.3.4 Testing of speedOfSound field

Field speedOfSound defines the initial delay introduced to sound when the source and the viewpoint are in different locations in the scene. The delay simulates the propagation delay of sound in a medium and is dependent on the distance between the source and the listener, and the speed of sound propagation in the medium. This field is tested both objectively and subjectively. This test is carried out for two different values of speedOfSound, and by positioning the source at two different distances from the **Viewpoint**. The delay that should be applied to sound is given in seconds by:

$$d = \frac{s}{speedOfSound},$$

where *speedOfSound* is the value given by the field of a same name, and *s* is the distance between the **Viewpoint** and the **DirectiveSound** location.

9.2.5.3.4.1 Scene configuration and field characteristics of DirectiveSound

In these tests, **DirectiveSound** fields useAirabs and spatialize are set to FALSE, the distance field is set to 0, and the speedOfSound field is set to 340 or 170 (depending on the test scene under consideration).

A visual object is associated to the **DirectiveSound** node with a help of a **Transform** node. **TimeSensor** and **PositionInterpolator** nodes are used for moving the source dynamically in the subjective tests.

9.2.5.3.4.2 Test scenes

Objective testing of speedOfSound:

AABper27-28	The speedOfSound is given a value 340, and in these two scenes the delay it causes is measured from the compositor output when replacing the sound source at 0 and 100 meter distance from the **Viewpoint**. The 0 distance must cause no delay to sound, and the delay at 100-meter distance should match that calculated from the equation in 9.2.5.3.4 with an accuracy of 10% or better. The delay is considered as the time lag (in addition to the lag introduced by the update interval of audio scene updates) in the response of the compositor to sound CU2_AAB_Px.
AABper29-30	Same as the test above, but CU4_AAB_Px is used as an input sound to the compositor.
AABper31-32	Same as AABper27-28 but with a value 170 given to speedOfSound.
AABper33-34	Same as above but for input sound CU4_AAB_Px.

Subjective testing of speedOfSound:

AABper35-36	Value of speedOfSound is 340 and 170 in these scenes, respectively. The sound source is moved from 100 distance to 0 distance and back to 100 distance. First the sound is at 100-meter distance for 5 seconds. Then it moves to 0 distance and back to 100 distance (simulating a source passing the listener) with a uniform speed during 10 s. The changing delay should be heard as a Doppler effect causing a raise in the pitch of sound when the source is getting closer to the listener, and a decrease in the pitch of sound when it is drawn away from the listener. The changing delay has to be interpolated between samples so that the Doppler effect is heard and no artifacts such as clicks are audible. The change in the pitch should be heard twice as strong in AABper36 than in AABper35. The input test signal is CU1_AAB_Px.
AABper37-38	The same scenes as for AABper35-36 but with input sound CU3_AAB_Px.

9.2.5.3.5 Testing of useAirabs field

The useAirabs field is used to enable distance dependent lowpass filtering caused by the stronger sound absorption of air at high frequencies than at low frequencies. The testing of this field is done both objectively and subjectively.

9.2.5.3.5.1 Scene configuration and field characteristics of DirectiveSound

In these tests, **DirectiveSound** field useAirabs is set to TRUE, and spatialize field is set to FALSE, the speedOfSound and the distance fields are set to 0.

A visual object is associated to **DirectiveSound** node with a help of a **Transform** node. **TimeSensor** and **PositionInterpolator** nodes are used for moving the source dynamically in subjective tests. Objective tests measure the level of the impulse response at three different distances between the source and the listener.

9.2.5.3.5.2 Test scenes

Objective testing of useAirabs:

AABper39-41	An impulse response of the compositor output is measured and a magnitude response is computed from it at distances 10, 50, and 100 meters between the **DirectiveSound** source and the **Viewpoint** in scenes AABper39-41. The magnitude response must match that computed from formula 5 in ISO 9613-1 at the given distance with an accuracy of 1 dB or better. The parameters concerning the atmospheric conditions are: humidity = 70%, temperature = 20 degrees centigrade, air pressure = 101325 Pa. The input sequence used in this test is CU2_AAB_Px.
AABper42-44	Same as the above test but with CU4_AAB_Px as the input source signal.

Subjective testing of useAirabs:

AABper45	In this test a **DirectiveSound** source is dynamically moving from a 1-meter distance to 100-meter distance. The air absorption filtering should be heard as increased lowpass filtering as a function of distance. No audible artifacts such as clicks should be heard as the filtering changes. The input sound used in this test is CU1_AAB_Px
AABper46	Same as the above test but with CU3_AAB_Px.

9.2.5.4 Procedure to test PerceptualParameters

Below is the node interface of **PerceptualParameters**

PerceptualParameters {

sourcePresence	1.0
sourceWarmth	1.0
sourceBrilliance	1.0
roomPresence	0.0
runningReverberance	1.0
envelopment	0.0
lateReverberance	1.0
heavyness	1.0
liveness	1.0
omniDirectivity	1.0
directFilterGains	1.0, 1.0, 1.0
inputFilterGains	1.0, 1.0, 1.0
refDistance	1.0
freqLow	250.0
freqHigh	4000.0
timeLimit1	0.02
timeLimit2	0.04
timeLimit3	0.1
modalDensity	0.8

}

The following nodes are used in the testing of **PerceptualParameters:**

Advanced AudioBIFS nodes:

DirectiveSound

PerceptualParameters

Other nodes used in the scenes:

Group

Viewpoint

DirectionalLight

Transform

Shape

Appearence

Material

Sphere

Cylinder

TimeSensor

OrientationInterpolator

9.2.5.4.1 Testing of the generic room response

This Subclause describes testing of **sourcePresence**, **roomPresence**, **runningReverberance**, **envelopment**, **lateReverberance**, **timeLimit1**, **timeLimit2**, **timeLimit3** and **modalDensity** fields.

The perceptual approach is based on the synthesis of a virtual room, the acoustic properties of which are based on a generic impulse response model. The time/frequency characteristics of the impulse response are derived from nine perceptual parameters associated with explicit time and frequency limits according to tables and equations of 7.2.2.93.2.1 and 7.2.2.93.2.2 of ISO/IEC 14496-11:2005.

9.2.5.4.1.1 Scene configuration and field characteristics of DirectiveSound and PerceptualParameters

In these tests the fields of the **DirectiveSound** node are set to default values, except that the roomEffect is set to TRUE, and spatialize field is set to FALSE for the objective tests. The fields the **PerceptualParameters** node are set to the default values except those that are tested (see list above).

9.2.5.4.1.2 Test scenes

Objective testing of the room response:

AABper47	This scene simulates a small room with no frequency dependent effects. The impulse response of the compositor is recorded. The delays and the levels of the recorded signal must match the corresponding values of the generic response according to the equations of 7.2.2.93.2.1 and 7.2.2.93.2.2 of ISO/IEC 14496-11:2005. The accuracy required for levels is 1 dB or better, and the accuracy required for the time limits, the decay time and the modal density is 10% or better. The levels R0, R1, R2 and R3, are calculated by summing the squared magnitudes of all the samples in the corresponding sections of the recorded impulse response. The reverberation time is measured (according to ISO 3382). The modal density is estimated by visual inspection of the power spectrum computed for section R3 in the impulse response, limited to a narrow frequency range around 1000 Hz, so as to evaluate the number of peaks per hertz in the frequency response. Input sound is CU2_AAB_Px.
AABper48	This scene simulates a large room with no frequency dependent effects. The impulse response of the compositor is recorded. The delays and the levels of the recorded signal must match the corresponding values of the generic response according to the equations of 7.2.2.93.2.1 and 7.2.2.93.2.2 of ISO/IEC 14496-11:2005. Accuracy requirements and measurement methods are identical to those of AABper47. Input sound is CU2_AAB_Px.
AABper49-50	Same procedure and field values as for AABper47 and AABper48, but with input sound CU4_AAB_Px.

Subjective testing of the room response:

AABper51	This scene simulates a small room. The parameters setting are the same as for AABper47. The sound should be perceived as in a small room. Input sound is CU1_AAB_Px.
AABper52	This scene simulates a large room. The parameters setting are the same as for AABper48. The sound should be perceived as in a large room. Input sound is CU1_AAB_Px.

AABper53-54 Same procedure and field values as for AABper51 and AABper52, but with input sound CU3_AAB_Px.

9.2.5.4.2 Testing of frequency-dependent effects

This Subclause describes testing of **sourceWarmth**, **sourceBrilliance**, **heaviness** and **liveness**.

9.2.5.4.2.1 Scene configuration and field characteristics of DirectiveSound and PerceptualParameters

In these tests the fields of the **DirectiveSound** node are set to default values, except that the roomEffect is set to TRUE, and spatialize field is set to FALSE. The fields the **PerceptualParameters** node are set to the default values except those that are tested (see list above) and the frequency limits which are modified according to the sampling rate.

9.2.5.4.2.2 Test scenes

Objective testing of frequency-dependent effects:

AABper55 This scene simulates a room with the same perceptual parameter values as in AABper48, except that sourceWarmth is set to 10.0 and sourceBrilliance is set to 0.1. The impulse response of the compositor is recorded and band-pass filtered around 1000 Hz with a bandwith of approximately one octave. The levels R0, R1, R2, R3 and the reverberation time are calculated in the same manner as in AABper47. The levels R1, R2, R3 relative to R0 must match with an accuracy of 1 dB the corresponding values according to the equations of 7.2.2.93.2.1 and 7.2.2.93.2.2 of ISO/IEC 14496-11:2005. The decay time must be matched with an accuracy of 10%. For both R0 and R1, the magnitudes at frequencies freqLow and freqHigh relative to the magnitude at 1000 Hz must match the sourceWarmth and sourceBrilliance values within 1 dB. Input sound is CU4_AAB_Px.

AABper56 This scene simulates a room with the same perceptual parameter values as in AABper55, except that freqLow and freqHigh are set respectively at 100 and 6000 Hz. The procedure is the same as for AABper55, except that the magnitude of R0 and R1 are evaluated at 100 and 6000 Hz instead of the default freqLow and freqHigh.

AABper57 Same as AABper56, but with CU2_AAB_Px as the input signal and freqHigh set at 3000 Hz.

AABper58 This scene simulates a room with the same perceptual parameter values as in AABper48, except that heaviness is set to 10.0 and liveness is set to 0.1. The impulse response of the compositor is recorded and band-pass filtered around 1000 Hz with a bandwith of approximately one octave. The levels R0, R1, R2, R3 and the reverberation time are calculated in the same manner as in AABper47. The levels R1, R2, R3 relative to R0 must match with an accuracy of 1 dB the corresponding values according to the equations of 7.2.2.93.2.1 and 7.2.2.93.2.2 of ISO/IEC 14496-11:2005. The reverberation time at 1000 z, freqLow and freqHigh must be matched with an accuracy of 10%. Input sound is CU4_AAB_Px.

AABper59 This scene simulates a room with the same perceptual parameter values as in AABper58, except that freqLow and freqHigh are set respectively at 100 and 6000 Hz. The procedure is the same as for AABper55, except that the reverberation time is evaluated at 100 and 6000 Hz instead of the default freqLow and freqHigh.

AABper60 Same as AABper59, but with CU2_AAB_Px as the input signal and freqHigh set at 3000 Hz.

Subjective testing of the frequency-dependent effects:

AABper61	This scene simulates a small room. The parameters setting are the same as for AABper47 except for the sourceWarmth and the sourceBrilliance that vary along with the time : 5 econds with default values, 5 seconds with sourceWarmth=10.0 and sourceBrilliance=0.1 and 5 seconds with sourceWarmth=0.1 and sourceBrilliance=10. The frequency-dependent effects should be perceived and no artifact should be heard during the changes of parameter settings. Input sound is CU1_AAB_Px.
AABper62	This scene simulates a large room. The parameters setting are the same as for AABper48 except for the heaviness and the liveness that vary along with the time : 5 econds with default values, 5 econds with heaviness=10.0 and liveness =0.1 and 5 econds with heaviness=0.1 and liveness =1.0. The frequency-dependent effects should be perceived and no artifact should be heard during the changes of parameter settings. Input sound is CU1_AAB_Px.
AABper63-64	Same procedure and field values as for AABper61 and AABper62, but with input sound CU3_AAB_Px.

9.2.5.4.3 Testing of InputFilterGains and directFilterGains

This Subclause describes testing of **directFilterGains** and **inputFiltergains** fields.

9.2.5.4.3.1 Scene configuration and field characteristics of DirectiveSound and PerceptualParameters

In these tests the fields of the **DirectiveSound** node are set to default values, except that the roomEffect is set to TRUE, and spatialize field is set to FALSE. The fields the **PerceptualParameters** node are set to the default values except those that are tested (see list above) and the frequency limits which are modified according to the sampling rate.

9.2.5.4.3.2 Test scenes

Objective testing of inputFiltergains and directFilterGains:

AABper65	This scene is the same as for AABper47, except that **inputFiltergains** is set to [0.1,0.1,0.1]. The impulse response of the compositor is recorded. The delays, the levels and the decay time of the recorded signal must be the same as in AABper47 except that the levels are reduced by 20 B.
AABper66	This scene is the same as for AABper47, except that **directFiltergains** is set to [0.1,0.1,0.1]. The impulse response of the compositor is recorded. The delays, the levels and the decay time of the recorded signal must be the same as in AABper47 except that the level of R0 (direct path) must be reduced by 20 B.
AABper67-68	Same procedure and field values as for AABper65 and AABper66, but with input sound CU4_AAB_Px.
AABper69	This scene is the same as for AABper47, except that **inputFiltergains** is set to [10.0,1.0,0.1]. The delays, the levels and the decay time of the recorded signal must be the same as in AABper47 except that the levels are increased by 20 dB at frequency freqLow and reduced by 20 B at frequency freqHigh.
AABper70	This scene is the same as for AABper47, except that **directFiltergains** is set to [10.0,1.0,0.1]. The delays, the levels and the decay time of the recorded signal must be the same as in AABper47 except that the direct-path (R0) levels are increased by 20 B at frequency freqLow and reduced by 20 B at frequency freqHigh.

AABper71-72 Same procedure and field values as for AABper69 and AABper70, but with input sound CU4_AAB_Px.

9.2.5.4.4 Testing of omnidirectivity

This Subclause describes testing of **omniDirectivity** field.

9.2.5.4.4.1 Scene configuration and field characteristics of DirectiveSound and PerceptualParameters

In these tests the fields of the **DirectiveSound** node are set to default values, except that the roomEffect is set to TRUE, and spatialize field is set to FALSE. The fields the **PerceptualParameters** node are set to the default values except those that are tested (see list above) and the frequency limits which are modified according to the sampling rate.

9.2.5.4.4.2 Test scenes

Objective testing of omniDirectivity:

AABper73 This scene is the same as for AABper47, except that omniDirectivity is set to 0.1. The impulse response of the compositor is recorded. The delays, the levels and the decay time of the recorded signal must be the same as in AABper47 except that the levels in the section R1, R2 and R3 are reduced by 20 dB.

AABper74 Same procedure and field values as for AABper73 but with input sound CU4_AAB_Px.

AABper75 This scene is the same as for AABper47, except that omniDirectivity is frequency dependent (expressed as three gain-frequency pairs). The impulse response of the compositor is recorded. The delays, the levels and the decay time of the recorded signal must be the same as in AABper47 except for the levels in the section R1, R2 and R3. The magnitude frequency responses computed for each of these sections should differ from those obtained in AABper47, by an amount matching, with an accuracy of 1 dB, the gain-requency pairs in the omniDirectivity field.

AABper76 Same procedure and field values as for AABper75 but with input sound CU4_AAB_Px.

9.3 AudioBIFS v3 Nodes

9.3.1 Introduction

This Subclause describes the conformance testing for the rendering and output of AudioBIFS v3 nodes, which are used for adding support for shaped sound sources (**WideSound** node), Ambisonics™ audio streams to two-and three-dimensional BIFS scenes. Furthermore they allow 3D sound in 2D visual scenes (**Transform3DAudio** node) and simplify the use of pre-defined audio effects (**AudioFXProto**s). With AudioBIFS v3 also a labelling mechanism was introduced that transports the channel configuration information through the AudioBIFS sub-graph and can be altered with the **AudioChannelConfig** node.

9.3.2 Composition Unit Inputs

The input audio streams used in the conformance testing of AudioBIFS v3 shall be outputs of an AAC decoder (AOT 2), and they are monophonic, stereophonic, 5.1 (surround) and Ambisonics™ sounds. They are explained below:

ABv3_CU01_2ch_AOT2_L1R2: Composition Unit Input AAC: white noise for one second on left, for one second on right channel. Duration: 8 seconds, sampling rate 44100 Hz, stereo.

ABv3_CU02_2ch_AOT2_L3R4: Composition Unit Input AAC: after 2 seconds silence white noise for one second on left, then for one second on right channel. Duration: 8 seconds, sampling rate 44100 Hz, stereo.

ABv3_CU03_2ch_AOT2_L5R6: Composition Unit Input AAC: after 4 seconds silence white noise for one second on left, then for one second on right channel. Duration: 8 seconds, sampling rate 44100 Hz, stereo.

ABv3_CU04_1ch_AOT2_M5: Composition Unit Input AAC: after 4 seconds silence white noise for one second. Duration: 8 seconds, sampling rate 44100 Hz, mono.

ABv3_CU05_1ch_AOT2_M6: Composition Unit Input AAC: after 5 seconds silence white noise for one second. Duration: 8 seconds, sampling rate 44100 Hz, mono.

ABv3_CU06_1ch_AOT2_M0-3: Composition Unit Input AAC: 3 seconds white noise starting from the beginning. Duration: 3 seconds, sampling rate 44100 Hz, mono.

ABv3_CU07_6ch_AOT2_surround: Composition Unit Input AAC: 0.5 seconds silence, then for 0.2 seconds an 880Hz sinus tone on center channel, 0.3s silence, 0.5s 440Hz sin on front left channel, 0.5s 440Hz sin on front right channel, 0.4s 220Hz sin on left surround channel, 0.1s silence, 0.4s 220Hz sin on right surround channel, 0.1s silence, 1s white noise on LFE channel. Duration: 4 seconds, sampling rate 44100 Hz, 5.1 channels.

ABv3_CU08_1ch_AOT2_one: Composition Unit Input AAC: male voice ‘one’. Duration: 1 second, sampling rate 44100 Hz, mono.

ABv3_CU09_1ch_AOT2_two: Composition Unit Input AAC: male voice ‘two’. Duration: 1 second, sampling rate 44100 Hz, mono.

ABv3_CU10_1ch_AOT2_three: Composition Unit Input AAC: male voice ‘three’. Duration: 1 second, sampling rate 44100 Hz, mono.

ABv3_CU11_1ch_AOT2_four: Composition Unit Input AAC: male voice ‘four’. Duration: 1 second, sampling rate 44100 Hz, mono.

ABv3_CU12_1ch_AOT2_five: Composition Unit Input AAC: male voice ‘five’. Duration: 1 second, sampling rate 44100 Hz, mono.

ABv3_CU13_1ch_AOT2_six: Composition Unit Input AAC: male voice ‘six’. Duration: 1 second, sampling rate 44100 Hz, mono.

ABv3_CU14_2ch_AOT2_LCR_Binaural: Composition Unit Input AAC: male voices ‘left, center, right’, repeated 1 time. Duration: 10 seconds, sampling rate 44100 Hz, 2-channel binaural.

ABv3_CU15_2ch_AOT2_applause: Composition Unit Input AAC: applause. Duration: ~12 seconds, sampling rate 44100 Hz, 2-channel stereo.

ABv3_CU16_1ch_AOT2_applause_mono: Composition Unit Input AAC: applause. Duration: ~12 seconds, sampling rate 44100 Hz, 1-channel mono.

ABv3_CU17_2ch_AOT2_orchestra: Composition Unit Input AAC: orchestra. Duration: ~12 seconds, sampling rate 44100 Hz, 2-channel stereo.

ABv3_CU18_1ch_AOT2_orchestra_mono: Composition Unit Input AAC: orchestra. Duration: ~12 seconds, sampling rate 44100 Hz, 1-channel mono.

ABv3_CU19_1ch_AOT2_W: Composition Unit Input AAC: the omni-directional channel (0^{th} order component W) of a 3D ambisonic recording of an ambient sound field: a complex, mostly static, outdoor scene; birds singing all around, splashes caused by stones thrown in the water first on the left then on the right, a child and adults speaking on the front, slightly on the left. In the test sequences, it has to be decoded and played synchronously with the following audio streams (ABv3_CU20_… to ABv3_CU22_...). Duration: ~14.3 seconds, sampling rate 44100 Hz, 1-channel.

ABv3_CU20_2ch_AOT2_XY: Composition Unit Input AAC: the horizontal, bidirectional (1^{st} order) channels X and Y of the same ambisonic recording as ABv3_CU19_1ch_AOT2_W. Duration: ~14.3 seconds, sampling rate 44100 Hz, 2-channels.

ABv3_CU21_1ch_AOT2_Z: Composition Unit Input AAC: the vertical bidirectional (1^{st} order) channel Z of the same ambisonic recording as ABv3_CU19_1ch_AOT2_W. Duration: ~14.3 seconds, sampling rate 44100 Hz, 1-channel.

ABv3_CU22_2ch_AOT2_UV: Composition Unit Input AAC: the horizontal, 2^{nd} order channels U and V of the same ambisonic recording as ABv3_CU19_1ch_AOT2_W. Duration: ~14.3 seconds, sampling rate 44100 Hz, 2-channels.

ABv3_CU23_1ch_AOT2_ambITU_C: Composition Unit Input AAC: the center channel of a 5.0 playback version of the recording described for ABv3_CU19_1ch_AOT2_W that could be played over a standard ITU loudspeaker arrangement. Duration: ~14.3 seconds, sampling rate 44100 Hz, 1 channel.

ABv3_CU24_2ch_AOT2_ambITU_LR: Composition Unit Input AAC: the front left and front right channels the 5.0 playback version described above. Duration: ~14.3 seconds, sampling rate 44100 Hz, 2 channels.

ABv3_CU25_2ch_AOT2_ambITU_SLSR: Composition Unit Input AAC: the surround left and surround right channels the 5.0 playback version described above. Duration: ~14.3 seconds, sampling rate 44100 Hz, 2 channels.

Abv3_CU26_2ch_AOT2_seawash_cl.media: Composition Unit Input AAC: center channel and left channel ‘seawash’. Duration: ~20 seconds, sampling rate 48000 Hz, 2 channels.

Abv3_CU27_2ch_AOT2_seawash_rs.media: Composition Unit Input AAC: right channel and surround channel ‘seawash’. Duration: ~20 seconds, sampling rate 48000 Hz, 2 channels.

Abv3_AudioNaturalReverb_ImpulseResponse.wav: Original impulse response for binary comparison for test scene Abv3_aFXP_aNa01.

9.3.3 Compositor Output

The output of the audio compositor will be investigated for conformance, and shall be a single output, N channel (depending on the spatialization and reproduction method used) PCM audio stream. The input audio streams are at 16 bit signed integer sample format, and the processing defined by the Advanced Audio BIFS nodes in the scene will be carried out at an accuracy of at least 16 bits.

Because of the non-normative nature of implementing many of the AudioBIFS features, no sample-wise comparison is done to the output sound from the compositor. Some of the features can be evaluated in a static situation (no dynamic changes, such as sound source or viewpoint movements, in the 3D environment) by measuring certain parameters of the compositor's output. Some functionalities, on the other hand, require testing in a dynamic situation where only subjective evaluation can be used (the user is listening to the sound compositor output, and watching the visual compositor output if visual components are present).

In addition to objective and subjective testing a third on one is needed for AudioBIFS v3, the parameter printout. Therefore the printout of certain parameters (like the absolute position of an audio source or the current channel configuration of the audio data) can be compared with a given reference. Note, that the

parameter printout does not print the *contents* of a node's ield, but the status information like channel configuration that has to be passed through the audio scene graph along with the audio data or through the BIFS scene graph like the transform hierarchy. The parameter printout should be ASCII text with the format

```
parameterName value_1 … value_n
```

whereby value should be in the format of the corresponding field and logged in the top-level sound nodes. In case of a SFFloat/MFFloat field type the 1.16 float format should be used.

9.3.4 Conformance Tests for AudioBIFS v3 Nodes

The following subclauses contain the detailed description of the conformance tests for the nodes **AdvancedAudioBuffer**, **AudioChannelConfig**, **SurroundingSound**, **Transform3Daudio**, **WideSound** and **AudioFXProto** effects.

9.3.4.1 Testing of AdvancedAudioBuffer Node

The **AdvancedAudioBuffer** node provides an interface for stored sound. This node has corrected functionality and enhanced reload mechanism compared to the **AudioBuffer** node, e.g to accumulate snippets of sound in the **AdvancedAudioBuffer**. These snippets can be accessed directly or as the full accumulated content.

9.3.4.1.1 BIFS components needed in the conformance testing

For testing the load and playback mechanism of the **AdvancedAudioBuffer** node a minimal set of BIFS nodes is needed: Besides the root node and one grouping node (**Group**, **OrderedGroup**) one top-level AudioBIFS node (**Sound**, **Sound2D**) as well as a node that connects the AudioBIFS sub-graph with the decoder (**AudioSource**) is required.

9.3.4.1.2 Conformance testing procedure

Conformance testing of the **AdvancedAudioBuffer** node requires the player to support the parameter printout of **AdvancedAudioBuffer** node's fields. For subjective testing the listener has to check if playing back the scene has the described effect.

The basic behaviour of this node is determined by the **loadMode** field. For each of these five modes a test scene shall be used to check the functionalities. The read/write access to the different content blocks shall be verified with the corresponding scene by a printout of the index of the content block. Additional subjective listening tests shall be performed.

Test Scenes:

ABv3_AAB01 This scene is used for testing the compatibility mode (**LoadMode**=0) (compatible to **AudioBuffer**, see functionality and semantics in the node definition) of the **AdvancedAudioBuffer** node by loading ABv3_CU8_1ch_AOT2_one with the help of the **AudioSource** node into the internal buffer of the **AdvancedAudioBuffer** node. After 10 seconds (t=10s) the 'one' will be repeated five times, which can be tested subjectively.

ABv3_ AAB02 This scene is used for testing the reload mode (**LoadMode**=1) by loading ABv3_CU08_1ch_AOT2_one with the help of the AudioSource node into the internal buffer of the AdvancedAudioBuffer node. The clip will be repeated five times with loop=enabled from t=2s until t=7s. After 10 seconds the content will be replaced be clip ABv3_CU08_1ch_AOT2_two, which will be repeated five times from t=12 until t=17. Clip ABv3_CU08_1ch_AOT2_three will be loaded after 20 seconds and repeated five times from t=22 until t=27. The playback should be tested subjectively.

ABv3_ AAB03 This scene is used for testing the accumulate mode (**LoadMode**=2) by loading ABv3_CU8_1ch_AOT2_one at t=0 with the help of the **AudioSource** node into the internal buffer of the **AdvancedAudioBuffer** node. At t=3s ABv3_CU08_1ch_AOT2_two will be loaded into the buffer and at t=6s ABv3_CU08_1ch_AOT2_three will be loaded into the buffer. At t=10s all three clips will be played contiuous ('one, two, three') two times while loop is enabled. The loop mechanism will be tested by setting loop to false at t=19s and start replay at t=20s for 6 seconds. Only one block ("one, two, three") should be heard due to the disabled loop mode. The playback should be tested subjectively.

ABv3_ AAB04 This scene is used for testing the continuous accumulate mode (**LoadMode**=3) by loading ABv3_CU08_1ch_AOT2_one, ABv3_CU08_1ch_AOT2_two and ABv3_CU08_1ch_AOT2_three at t=0, t=3s and t=6s into the internal buffer of the **AdvancedAudioBuffer** node with the help of **AudioSource**. At t=10s all three clips will be played contiuous ("one, two, three") two times while loop is enabled. At t=18s a fourth clip (ABv3_CU08_1ch_AOT2_four) will be appended and with the restriction of **length** = 3 seconds, the first clip shall be deleted. At t=20s all three clips will be played contiuously ("two, three, four") two times. This sequence will be repeated with appending ABv3_CU08_1ch_AOT2_five and playing back ("three, four, five") at t=30s and appending ABv3_CU08_1ch_AOT2_six and playing back ("four, five, six") at t=40s. The playback should be tested subjectively.

ABv3_AAB05 This scene is used for testing the accumulate mode with limited number of buffer blocks of the **AdvancedAudioBuffer** node. In this mode the number of accumulaed blocks will be set instead of a maximum length. First a sequence "one, two, three" will be loaded and will be played back two times at t=10s. Then "four" will be loaded into the buffer by replacing "one", played back two times at t=20s. At t=27s the block in the middle ("three") will be deleted and at t=29 "five" will be appended. This block ("two, four, five") will be played back two times at t=31s.

At t=40s the latest block ("five") is individually addressed and shall be played 3 times while loop is still enabled. At t=43s the block in the middle ("four") will be addressed individually and played back 3 times. At t=46s the first block ("two") will be addressed individually and played back 3 times until t=49s. At t=48,5s **loop** will be set to false. The actual active block will be played until its end.

At t=50s, t=55s and t=60s clip "four", "five" and "two" will be played only one time.

The playback should be tested subjectively.

9.3.4.2 Testing of AudioChannelConfig Node

This node is used to label the audio data in the audio subtree to supply the audio presenter with the required information for multi-channel or soundfield signals. The node has the standard audio node interfaces, but no signal processing capability. The samples are passed through and get new channel configuration information.

9.3.4.2.1 BIFS components needed in the conformance testing

For testing the labelling mechanism of the **AudioChannelConfig** node a minimal set of BIFS nodes is needed: Besides the root node and one grouping node (**Group**, **OrderedGroup**) one top-level AudioBIFS node that has a **spatialize** field (**Sound**, **Sound2D**, **DirectiveSound**) as well as a node that connects the AudioBIFS sub-graph with the decoder (**AudioSource**) is required.

9.3.4.2.2 Conformance testing procedure

Conformance testing of the **AudioChannelConfig** node requires the player to support the parameter printout of the channel configuration that reaches the top-level sound nodes. As the way this information is transported through the AudioBIFS sub-graph is implementation-dependent the channel configuration should be printed in the form of the **AudioChannelConfig** node's fields.

9.3.4.2.2.1 Unique generalChannelFormat

In this Subclause the testing of the correct composition of one or more audio signal of the same generalChannelFormat is described, i.e. all sources in the scene are channel-oriented presets (or subsets thereof), parametric channel oriented or Ambisonics™ oriented.

Pass through

Test scene for parameter printout and subjective testing:

Abv3_ACC01 This scene is used for testing whether the channel configuration of the audio elementary stream is correctly passed through the AudioBIFS tree. For that purpose a 5.1-channel standard multi-channel configuration sound source will be used. The audio compositor shall recognize the channel configurations of the elementary stream and map the channels to the appropriate speaker(s). The 5.1 configuration shall be recognizable in the *parameter printout* of the channel configuration from the top level node.

For *subjective testing* the CU input used for this scene result in a distinct scheme: On every channel white noise is being played for one second, starting with the front left channel, followed by the front right, surround left, surround right, center channel and finally the LFE channel (5s<t<6s).

ABv3_CU07_6ch_AOT2_surround is used as input sounds for this scene.

ChannelPreset

Abv3_ACC02 This scene is used for testing whether an alternative channel configuration stream will be passed correctly through the AudioBIFS tree. Therefore **generalChannelFormat** will be set to the “ChannelPreset” mode. A binaural recorded sequence will be used. After start-up the clip will be marked as “normal” stereo. At t=12s the clip will be played again and marked as “binaural” stereo. The configuration shall be recognizable in the *parameter printout* of the channel configuration from the top level node.

For *subjective testing* the properties of the audio presenter must be taken into account. The presenter might have a crosstalk-canceller for loudspeaker playback or a HRTF cross-convolution for headphone playback. In this case the clip will be heard ‘spatialized’ false during the first 12 seconds and for t>=12s the clip will be heard correctly due to the binaural marked content which should lead to the crosstalk-canceller being enabled for loudspeaker listening or disabled HRTF cross-convolution in the headphone listening case.

ABv3_CU08_1ch_AOT2_LCR_Binaural is used as input sounds for this scene.

ChannelPresetSubset

Test scene for parameter printout and subjective testing:

ABv3_ACC03 This scene is used for testing the combination of two stereo and two mono sound sources to a 5.1-channel sound source. The audio compositor shall recognize the channel configurations of the sound sources and map the channels to the appropriate speaker(s). It shall be recognizable in the *parameter printout* of the channel configuration that each top-level sound node addresses an individual subset of a 5.1 configuration (the values of **fixedPreset** are 000110b=6 for the front channels, 011000b=24 for the surround

channels, 000001b=1 for the center channel and 100000b=32 for the LFE channel).

For *subjective testing* the CU inputs used for this scene result in a distinct scheme: On every channel white noise is being played for one second, starting with the front left channel, followed by the front right, surround left, surround right, center channel and finally the LFE channel (5s<t<6s).

ABv3_CU01_2ch_AOT2_LR12, ABv3_CU02_2ch_AOT2_LR34, ABv3_CU04_1ch_AOT2-_M5 and ABv3_CU05_1ch_AOT2_M6 are used as input sounds for this scene.

ABv3_ACC04 This scene is used for testing the combination of two mono sources to one stereo source. The audio compositor shall recognize the channel configurations of the sound sources and map the channels to the appropriate speaker. It shall be recognizable in the *parameter printout* of the channel configuration that each top-level sound node addresses an individual channel (the values of **fixedPreset** are 01b=1 for the left channel and 10b=2 for the right channel).

For *subjective testing* the CU inputs used for this scene result in a distinct scheme: On the left channel white noise is being played for three seconds, followed 1 second silence and then one second white noise on the right channel.

ABv2_CU06_1ch_AOT2_M0-3 and ABv3_CU04_1ch_AOT2_M5 are used as input sounds for this scene.

Abv3_ACC05 This scene is used for testing whether a dynamic changed channel configuration stream (relabelled channels) will be passed correctly through the AudioBIFS subtree. Therefore **generalChannelFormat** will be set to the “ChannelPresetSubset” mode. A stereo signal (“applause”) will be used. After start-up the clip will be marked as L,R signal. At t=6s the clip will be marked as LS, RS. The configuration shall be recognizable in the *parameter printout* of the channel configuration from the top level node.

For *subjective testing* the clip should be recognized from the left and right loudspeakers after start-up. After 6 seconds the clip should be heard from the left and right surround loudspeakers.

Abv3_Cu15_2ch_Aot2_Applause is used as input sounds for this scene.

ParametricChannelOriented

These scenes are used for testing whether an alternative channel configuration stream will be passed correctly through the AudioBIFS tree. Therefore **generalChannelFormat** will be set to the “ParametricChannelOriented” mode. The audio presenter shall recognize the loudspeaker positions of the sound sources and map the channels to the appropriate speaker(s). Therefore the ITU 5.1 standard configuration will be addressed (r=2m, 0° (center), +/-30° (right/left), +/-110° (right/left surround) and +15°azimuth/-21°elevation (LFE)). At t=12s the configuration will be flipped to (0°, -/+30°, -/+110° and +15°azimuth/-21°elevation). The configuration shall be recognizable in the *parameter printout* of the channel configuration from the top level node.

For *subjective testing* the, the left/right and left/right surround channels will be swapped at t=12s.

ABv3_CU7_6ch_AOT2_surround is used as input sounds for scene Abv3_ACC06a … Abv3_ACC07c and Abv3_ACC09.

Abv3_ACC06a origin = scene origin, Cartesian corrdinate system

Abv3_ACC06b origin = scene origin, Polar coordinate system

Abv3_ACC06c	origin = scene origin, Cylindric coordinate system
Abv3_ACC07a	origin = user position, Cartesian corrdinate system
Abv3_ACC07b	origin = user position, Polar coordinate system
Abv3_ACC07c	origin = user position, Cylindric coordinate system
Abv3_ACC09	[C, Left, Right, Left Surround, Right Surround, LFE] = [scene origin: Cartesian, Polar, Cylindric, user position: Cartesian, Polar, Cylindric]

ParametricChannelOriented with directional Loudspeaker

Abv3_ACC10 This scenes is used for testing loudspeaker directional parameters. The **generalChannelFormat** will be set to the "ParametricChannelOriented" mode. The audio presenter shall recognize the loudspeaker positions of the sound sources and map the channels to the appropriate speakers. A 4-channel subset of the ITU 5.1 standard configuration with only one surround channel will be addressed (r=2m, 0° (center), +/-30° (right/left), -180° (dipole surround)), similar to a Dolby ProLogic® configuration. The surround channel will be set to dipole characteristic with **channelDirectionalPattern** = 1. The configuration shall be recognizable in the *parameter printout* of the channel configuration from the top level node.

For *subjective testing*, the surround channel should give a surround feeling, due to the missing lobe from the surround loudspeaker.

origin = user position, Cartesian corrdinate system; Abv3_CU26_2ch_AOT2_sea-wash_cl and Abv3_CU26_2ch_AOT2_seawash_rs are used as input sounds for the scene. The two 2-channel streams are combined to a 4-channel signal, whereby the 4th channel the surround channel is.

ParametricAmbisonicsOriented

ABv3_ACC11 This scene is used for testing the gathering of two stereo and two mono streams into a multi-channel audio flow labelled as a Higher Order Ambisonic sound field. In this scene, the input ambisonic channels are in the default order (W, X, Y, Z, U, V) regarding the specification of the spatial resolution (**ambResolution2D** and **ambResolution3D**) and therefore they follow an implicit indexing. The audio compositor shall recognize the channel configuration of the output multi-channel flow. It shall be recognizable in the *parameter printout* of the channel configuration that the top-level **SurroundingSound** sound node addresses an Ambisonic sound field with a 2nd order 2D resolution (*i.e.* regarding the horizontal plane) and a 1st order 3D resolution (*i.e.* regarding the vertical dimension):

```
SurroundingSound: generalChannelFormat 4
SurroundingSound: ambResolution2D 2
SurroundingSound: ambResolution3D 1
SurroundingSound: ambEncodingConvention 2
```

For *subjective testing*: The 3D audio compositor should take into account the channel configuration of the output multi-channel flow in the most appropriate way to process its spatial decoding and render the sound field according to the reproduction setup (a loudspeaker setup or headphones). The listener should perceive the same spatial scene organization as described when presenting the Composition Unit ABv3_CU19_1ch_AOT2_W in 9.3.2

ABv3_CU19_1ch_AOT2_W, ABv3_CU20_2ch_AOT2_XY, ABv3_CU21_1ch_AOT2_Z and ABv3_CU22_2ch_AOT2_UV are used as input sounds for this scene.

ABv3_ACC12 This scene is similar to ABv3_ACC11 and uses the same input sounds except that it changes their order by permuting two children of the **AudioChannelConfig** node. Therefore an explicit indexing of Ambisonic channels is necessary and done by using the **ambComponentIndex** field. The parameter printout should make appear the following lines:

```
SurroundingSound: generalChannelFormat 4
SurroundingSound: ambResolution2D 2
SurroundingSound: ambResolution3D 1
SurroundingSound: ambEncodingConvention 2
SurroundingSound: ambArrangementRule 1
SurroundingSound: ambComponentIndex [0 1 2 4 5 3]
```

For subjective testing: from the listener's point of view, the rendered sound field should sound the same as with scene ABv3_ACC11.

For objective testing: output signals delivered by the audio compositor for restitution over loudspeakers or headphones should be the same as with scene ABv3_ACC11.

ABv3_ACC13 This scene has similarities with ABv3_ACC12 except that the first input flows (representing Ambisonic channels W, X, Y, U and V) are replaced with input streams forming a 5.0 content that can be reconverted into the Ambisonic channels W, X, Y, U and V of scenes ABv3_ACC11 (and ABv3_ACC12). This conversion is done by applying the matrix described by **ambBackwardMatrix** to the input channels. Moreover the **ambComponentIndex** field is used for explicit indexing of resulting Ambisonic channels.

The parameter printout should make appear the following lines:

```
SurroundingSound: ambResolution2D 2
SurroundingSound: ambResolution3D 1
SurroundingSound: ambEncodingConvention 2
SurroundingSound: ambArrangementRule 7
SurroundingSound: ambComponentIndex [0 1 2 4 5 3]
SurroundingSound: ambBackWardMatrix [6 6 0.1535 0.9007 0.9007
0.6796 0.6796 0 0.5075 1.4697 1.4697 -1.0610 -1.0610 0 0 0.5580
-0.5580 2.0493 -2.0493 0 8.4222 -4.1298 -4.1298 0.8123 0.8123 0
0 4.0084 -4.0084 -3.5519 3.5519 0 0 0 0 0 0 1]
```

For subjective testing: from the listener's point of view, the rendered sound field should sound substantially the same as with scene ABv3_ACC11 in terms of spatial organization.

ABv3_CU23_1ch_AOT2_ambITU_C, ABv3_CU24_2ch_AOT2_ambITU_LR, ABv3_CU25_2ch_AOT2_ambITU_SLSR and ABv3_CU21_1ch_AOT2_Z are used as input sounds for this scene.

9.3.4.2.2.2 Combination of different formats

To test the ability of the audio compositor to render several multi-channel formats in the same scene and at the same time, use the scene ABv3_SS04, which combines the same ambisonic sound field as used in scene ABv3_ACC11 and the same 5.1 sound field as used in scene ABv3_ACC01.

9.3.4.3 Testing of SurroundingSound Node

The **SurroundingSound** node is used to attach sound to a scene. This causes spatial qualities and makes it related to the visual content of the scene. This includes multi-channel signals that cannot be spatially transformed with the other sound nodes due to their restrictions to the specification of the **phaseGroup** field and the **spatialize** field. By using this node, sound may be attached to a group and spatialized or moved around as appropriate for the spatial transforms above the node.

9.3.4.3.1 BIFS components needed in the conformance testing

Except from basic 3D nodes (**Group**, **Transform**, **ViewPoint**) the **AdvancedAudioBuffer** node is needed for looped playback of the audio data. For subjective testing the scene needs to be animated. The nodes **TimeSensor, CoordinateInterpolator** and **OrientationInterpolator** are required as well as the ROUTE mechanism. To ensure the correct channel configuration an **AudioChannelConfig** node is used in the test scenes.

9.3.4.3.2 Conformance testing procedure

Conformance testing of the **SurroundingSound** node requires the player to support the parameter printout of **SurroundingSound** node's fields. For subjective testing the listener has to check if playing back the scene has the described effect. The subjective listening tests shall be performed.

Test Scenes:

ABv3_SS01 This scene is used for subjective testing of the **SurroundingSound** node. This is done by rotating a 5.1 audio track around the listener by modifying the **orientation** field of the node **SurroundingSound** via **TimeSensor** and **OrientationInterpolator** nodes. The listener shall perceive the clockwise rotation of the surround panorama.

ABv3_CU07_6ch_AOT2_surround is used as input sound for this scene.

ABv3_SS01b Same as ABv3_SS01, except that ABv3_CU23_1ch_AOT2_ambITU_C, ABv3_CU24_2ch_AOT2_ambITU_LR, ABv3_CU25_2ch_AOT2_ambITU_SLSR and ABv3_CU19_1ch_AOT1_W are used as input sounds for this scene (recording of a natural sound scene).

ABv3_SS02 This scene is used for subjective testing of the effect of outer transformations on the **SurroundingSound** node. This is done by rotating a 5.1 audio track around the listener by modifying the **rotation** field of an "outer" **Transform** node via **TimeSensor** and **OrientationInterpolator** nodes. The listener shall perceive the clockwise rotation of the surround panorama (similarly as with scene ABv3_SS01) as long as the **isTransformable** field of the **SurroundSound** node is set to TRUE, *i.e.* during the first 8 seconds. After this delay the **isTransformable** field is set to FALSE so that the **rotation** field of the **Transform** node shall have no longer effect on the **SurroundingSound** node and the listener shall perceive a static surround panorama.

ABv3_CU07_6ch_AOT2_surround is used as input sound for this scene.

ABv3_SS02b Same as ABv3_SS02, except that ABv3_CU23_1ch_AOT2_ambITU_C, ABv3_CU24_2ch_AOT2_ambITU_LR, ABv3_CU25_2ch_AOT2_ambITU_SLSR and ABv3_CU19_1ch_AOT1_W are used as input sounds for this scene (recording of a natural sound scene).

ABv3_SS04 This scene is used for subjective testing of the distance attenuation, *i.e.* the decreasing (resp. increasing) of the perceived surrounding sound field level as the **ListeningPoint** moves away from (resp. gets closer to) its centre, as defined by the **location** field of the **SurroundingSound** node. In this scene, two sound fields (and

SurroundingSound instances) are present and centered on distinct locations. The **distance** parameter of the two **SurroundingSound** instances is chosen so that both sound fields are audible with a non negligible level half-way between the two locations. As **ListeningPoint** moves from one location to the other (during the first 10s), the listener shall first perceive the first surrounding sound field (nature ambience) with a significantly higher level than the second one, then a mix between both sound fields and finally mostly the second sound field (test signals). Then **ListeningPoint** moves back to its initial position so that the listener shall perceive the inverse effect (during the following 10 s). The kind of cross-fade effect implied by the distance attenuation should be as smooth as possible. For the next rounds the distance attenuation is deactivated for the first **SurroundingSound** instance (its **distance** parameter is set to 0) so that only the second sound field shall be perceived with a varying level as **ListeningPoint** continues its back-and-forth trajectory.

The sound field already used in scenes ABv3_AAC11 (2nd order ambisonic recording of nature ambience) and ABv3_AAC01 (5.1 test signals) are combined and respectively centered on the first and second location points. Therefore the present scene is also used for subjective testing of the combination of several formats.

ABv3_SS05

This scene is used for subjective testing of the effect the angular distortion of the sound scene as the **ListeningPoint** (or **ViewPoint**) moves in the virtual scene and when the **distortionFactor** is not null. It is also used to test the ability of the 3D audio compositor to appropriately combine several kinds of sound field transformations, namely angular distortion and rotation.

The same sound field as ABv3_AAC11 is used (ABv3_CU19_1ch_AOT2_W, ABv3_CU20_2ch_AOT2_XY, ABv3_CU21_1ch_AOT2_Z and ABv3_CU22_2ch_AOT2_UV are also used as input sounds). The sound field extract is 14.3 second long and played in loop using an **AdvancedAudioBuffer** node. The scene is divided in 5 sequences that correspond to several kinds of **ListeningPoint** movements and induced sound field transformations. The **distortionFactor** is set to 0.2, which corresponds to a reference distance of 5 meters for sounds positioned orthogonally to the **ListeningPoint**'s movement.

1st step (0→14.3 sec.): **ListeningPoint** is static, positioned and oriented according to the default parameter values. As the sound field contains mostly static sources, the listener shall be able to memorize its spatial organization.

2nd step (14.3→28.6 sec.): **ListeningPoint** rotates by half-turn during the first 7.15s. In the first 7.15s the listener should perceive that the sound scene rotates without changing the relative angles between the sounds of the scene, nor their level. In the last 7.15s the listener shall perceive that the front-back and left-right axes have been inverted with respect to the original orientation.

3rd step (28.6→42.9 sec.): **ListeningPoint** moves 5 meters towards its new frontal direction. The listener shall perceive that the front part of the scene progressively enlarges while the lateral sources move backward and the back part of the scene narrows. Changes should be rendered as smoothly as possible.

4th step (42.9→57.2 sec.): **ListeningPoint**'s movement combines half-turn rotation and diagonal trajectory so that its final position is 5m left to the initial position and its final orientation is the original one. The listener should feel both rotation and angular distortion effects. Changes should be rendered as smoothly as possible and without audible click.

5th step (57.2→71.5 sec.): During the first 7.15 s **ListeningPoint** moves to its right (like a crab) to get its initial position. The listener should perceive that the front and back sound sources move to left. He/she should finally retrieve the initial spatial organization.

9.3.4.4 Testing of Transform3DAudio Node

The **Transform3DAudio** node is used to integrate 3D audio in 2D visual scenes. Therefore coordinate transformations can be applied so that movements in the x-y (visual) plane are transformed into movements in the x-z plane for audio (which is the default behaviour of this node). Additionally it allows adding coordinates and vectors to the incoming two-dimensional transform hierarchy to achieve a full 3D addressing.

9.3.4.4.1 BIFS components needed in the conformance testing

For testing the **Transform3DAudio** node a minimal set of BIFS nodes is needed: Besides the root node and a node for 2D grouping (**OrderedGroup**) and transformations (**Transform2D**) one top-level 3D AudioBIFS node having a **spatialize** field (**Sound**, **DirectiveSound**) as well as a node that connects the AudioBIFS sub-graph with the decoder (**AudioSource**) is required.

9.3.4.4.2 Conformance testing procedure

Conformance testing of the **Transform3DAudio** node requires the player to support the parameter printout of the location and orientation of the respective **Sound** node in world coordinates (`globalPosition` and orientation `globalOrientation`). Note that the transformation produced by the **location** field of the **Sound** node also shall be taken into account for parameter printout. With this method the default functionality (flipping coordinate system), the coordinate transformation and the adding of coordinates and vectors to 2D transformations can be tested. For subjective testing the listener has to check if playing back the scene has the described effect.

Test Scenes:

ABv3_T3DA01 This scene is used for testing the default functionality of the **Transform3DAudio** node (mapping the x-y position into the x-z plane) by modifying the **translation** field of an "outer" **Transform2D** node via BIFS updates, which causes the sound source to move in the x-z plane. In the *parameter printout* it shall be visible that the global position in 3D world coordinates of the **Sound** node are initially (-1,0,0), and after each BIFS update (each 500ms) (0.707,0,-0.707), (1,0,0), (0.707,0,0.707) and (0,0,1):

```
Sound: globalPosition -1.0000 0.0000 0.0000
Sound: globalPosition 0.7071 0.0000 -0.7071
Sound: globalPosition 1.0000 0.0000 0.0000
Sound: globalPosition 0.7071 0.0900 0.7071
Sound: globalPosition 0.0000 0.0000 1.0000
```

For *subjective testing* the sound source shall describe a rough semicircle from its initial position right in front of the listener over the right side up to the back of the listener remaining in the x-z plane (y=0).

ABv3_T3DA02 This scene is used for testing the extension of a 2D rotation to a 3D rotation by adding a rotation vector (0,0,-1) (field **rotationVector**) to the rotation angle obtained from the 2D transform hierarchy in the **Transform3DAudio** node. The **rotationAngle** field of an "outer" **Transform2D** node is modified by BIFS updates, which cause the sound source to move in the x-z plane from the front of the listener to his right side (after the default coordinate transformation of the **Transform3DAudio** node is applied). In the parameter printout this movement shall be visible; the absolute position of the sound node changes each 500ms:

```
Sound: globalPosition 0.0000 0.0000 -1.0000
Sound: globalPosition 0.3827 0.0000 -0.9239
Sound: globalPosition 0.7071 0.0000 -0.7071
Sound: globalPosition 0.9239 0.0000 -0.3827
Sound: globalPosition 1.0000 0.0000 -0.0000
```

ABv3_T3DA03 This scene is used for testing the extension of a 2D rotation center to a 3D rotation center by adding a 3rd center coordinate 1 (field **thirdCenterCoordinate**) to the 2D rotation center (1,0) obtained from the 2D transform hierarchy in the **Transform3DAudio** node. The **rotationAngle** field of an "outer" **Transform2D** node is modified by BIFS updates, which cause the sound source to rotate around the transformed rotation center (after the default coordinate transformation of the **Transform3DAudio** node is applied). Additionally the **rotationVector** is (0,0,-1). In the parameter printout this circular movement shall be visible; the absolute position of the sound node changes each 375ms:

```
Sound: globalPosition 0.0000 1.0000 0.0000
Sound: globalPosition 1.0000 1.4142 0.0000
Sound: globalPosition 2.0000 1.0000 0.0000
Sound: globalPosition 2.4142 0.0000 0.0000
Sound: globalPosition 2.0000 -1.0000 0.0000
Sound: globalPosition 1.0000 -1.4142 0.0000
Sound: globalPosition 0.0000 -1.0000 0.0000
Sound: globalPosition -0.4142 0.0000 0.0000
Sound: globalPosition 0.0000 1.0000 0.0000
```

ABv3_T3DA04 This scene is used for testing the extension of a 2D scaling to a 3D scaling by adding a 3rd scale coordinate 3 (field **thirdScaleCoordinate**) to the 2D scaling (1,0) obtained from the 2D transform hierarchy in the **Transform3DAudio** node. After the default coordinate transformation of the **Transform3DAudio** node is applied the **location** vector of the **Sound** node (1,1,1) shall be (2,3,-2), which can be seen in the parameter printout (no BIFS updates are used, so only one value will be shown):

```
Sound: globalPosition 2.0000 3.0000 -2.0000
```

ABv3_T3DA05 This scene is used for testing the extension of an angular 2D scale orientation to a 3D scale orientation by adding the vector (0,1,0) (field **scaleOrientationVector**) to the 2D scale orientation obtained from the 2D transform hierarchy in the **Transform3DAudio** node. The **scaleOrientation** field of an "outer" **Transform2D** node is modified by BIFS updates, which cause the sound source to move in a plane parallel to the xy-plane (after the default coordinate transformation of the **Transform3DAudio** node is applied). A scaling of (2,2,3) ((2,2) from the 2D transform hierarchy and 3 as **thirdScaleCoordinate**) is applied in order to make the scale orientation effective. In the parameter printout this circular movement shall be visible; the absolute position of the sound node changes each 750ms:

```
Sound: globalPostion 2.0000 3.0000 -2.0000
Sound: globalPostion 3.0000 3.0000 -2.0000
Sound: globalPostion 3.0000 2.0000 -2.0000
Sound: globalPostion 2.0000 2.0000 -2.0000
Sound: globalPostion 2.0000 3.0000 -2.0000
```

ABv3_T3DA06 This scene is used for testing the extension of a 2D translation to a 3D translation by adding a 3rd translation coordinate (field **thirdTranslationCoordinate**) to the translation (1,1) obtained from the 2D transform hierarchy in the **Transform3DAudio** node. The **thirdTranslationCoordinate** field of the **Transform3DAudio** node is modified by BIFS updates, which cause the sound source to move from (1,0,-1) to (1,1,-1) (after the default coordinate transformation of the **Transform3DAudio** node is applied). In the parameter printout this movement shall be visible; the absolute position of the sound node changes each 500ms:

```
Sound: globalPosition 1.0000 0.0000 -1.0000
Sound: globalPosition 1.0000 0.2000 -1.0000
Sound: globalPosition 1.0000 0.4000 -1.0000
Sound: globalPosition 1.0000 0.6000 -1.0000
Sound: globalPosition 1.0000 0.8000 -1.0000
Sound: globalPosition 1.0000 1.0000 -1.0000
```

ABv3_T3DA07 This scene is used for testing the coordinate transform functionality of the **Transform3DAudio** node by modifying the axis of its **coordinateTransform** field via BIFS updates each 500ms, which causes the sound source to move. In the *parameter printout* these position changes shall be visible:

```
Sound: globalPosition 0.0000 0.0000 1.0000
Sound: globalPosition -0.0073 0.0370 0.9993
Sound: globalPosition -0.0451 0.1212 0.9916
Sound: globalPosition -0.1087 0.2093 0.9718
Sound: globalPosition -0.1789 0.2807 0.9430
Sound: globalPosition -0.2440 0.3333 0.9107
```

ABv3_T3DA08 This scene is used for testing the coordinate transform functionality of the **Transform3DAudio** node by modifying the angle of its **coordinateTransform** field via one BIFS update, which causes the sound source to flip from the xz-plane to the xy-plane. In the *parameter printout* this position change shall be visible:

```
Sound: globalPosition 0.0000 0.0000 -1.0000
Sound: globalPosition 0.0000 1.0000 0.0000
```

For all scenes ABv2_CU06_1ch_AOT2_M0-3 is used as input sound.

9.3.4.5 Testing of WideSound Node

The **WideSound** node is used to attach sound to a scene, thereby giving it spatial qualities with a determinable widening for non-phase-related signals from its descendant audio nodes and relating it to the visual content of the scene. Additionally the node has backward compatible functions of the **Sound** and **DirectiveSound** nodes which will not be tested here.

9.3.4.5.1 BIFS components needed in the conformance testing

For testing the **WideSound** node a minimal set of BIFS nodes is needed: Besides the root node and one grouping node (**Group**, **OrderedGroup**) one a node that connects the AudioBIFS sub-graph with the decoder (**AudioSource**) is required.

9.3.4.5.2 Conformance testing procedure

Conformance testing of the **WideSound** node requires the player to support the parameter printout of **WideSound** node's fields. For subjective testing the listener has to check if playing back the scene has the described effect.

Four different shapes of a sound can be selected: shuck, box, ellipsoid and cylinder. The printout of the nodes gives a functional hint. But the real function of the node can be tested subjectively only. Therefore it is a common practice to play back the same sound (here the four shapes) with different behaviour consecutively without disturbing noise. The subjective listening tests shall be performed.

9.3.4.5.2.1 Testing of all shapes

Test Scenes:

ABv3_WS01 This scene is used for testing the different shapes of the **WideSound** node, starting with a stereo version of the clip: ABv3_CU14_2ch_AOT2_applause. At t=15s, t=30s, t=45s and t=60s the clip ABv3_CU15_1ch_AOT2_applause_mono will be used with the four shapes consecutively shuck, box, ellipsoid and cylinder. The size of the shape is {2m,1m,1m} for the box, ellipsoid and cylinder in the location {0, 2, 0} and half/quarter sphere for the shuck.

ABv3_WS02 This scene is used for testing the different shapes of the **WideSound** node, starting with a stereo version of the clip: ABv3_CU17_2ch_AOT2_orchestra. At t=15s, t=30s, t=45s and t=60s the clip ABv3_CU18_1ch_AOT2_orchestra_mono will be used with the four shapes consecutively shuck, box, ellipsoid and cylinder. The size of the shape is {2m,1m,1m} for the box, ellipsoid and cylinder in the location {0, 2, 0} and half/quarter sphere for the shuck.

9.3.4.5.2.2 Testing of signalling plane waves

ABv3_WS03 This scene is used for testing the signalling of plane waves with the **WideSound** node. Therefore a source emanating two-dimensional plane waves is positioned in front of the listener. If the playback system is capable of rendering plane waves the sound source shall be perceived in infinite distance (meaning always from the front, independent from translation in x-z plane, *subjective testing*). After three seconds the source is switched to emanate three-dimensional plane waves. The listener shall perceive the source in infinite distance regardless of any translation.

ABv3_CU18_1ch_AOT2_orchestra_mono is used as sound input for this scene.

9.3.4.6 Testing of AudioFXProto mechanism

The **AudioFxPROTO** node provides an implementation of a tailored subset of functionality available through the **AudioFX** node. The **AudioFX** node normally requires a Structured Audio implementation. The tailored subset allows players without Structured Audio capability to use these standard audio effects.

The intended effect of the **AudioFxPROTO** node is not normative, so that the effect can be tested subjectively only. The **AudioFxPROTO** node interface is normative. The parameters shall be verified with the corresponding scene by a printout of the **audioFXParams** field.

9.3.4.6.1 BIFS components needed in the conformance testing

For testing the **AudioFXProto** with the **AudioFX** node a minimal set of BIFS nodes is needed: Besides the root node and one grouping node (**Group**, **OrderedGroup**) one a node that connects the AudioBIFS sub-graph with the decoder (**AudioSource**) is required. The effect can be implemented in a loadable module, addressed by the PROTO's name instead of a full **AudioFX** implementation.

9.3.4.6.2 Conformance testing procedure

Conformance testing of the **AudioFXProto** with the **AudioFX** node requires the player to support the parameter printout of the **audioFXParams** field. For subjective testing the listener has to check if playing back the scene has the described effect. The effects are tested with Abv3_CU17_1ch_AOT2_orchestra or Abv3_CU18_1ch_AOT2_orchestra_mono, depending on the effect. The effect parameters are changed during the playback of the clip.

9.3.4.6.2.1 Testing of audioChorus

Abv3_aFXP_aCh01 This scene is used for testing the **audioChorus** effect. At start-up the effect is disabled (bypass of the clip). At t=6s the parameters inside the **audioFXParams** field are set to:

`rate=2.0, depth=0.5` and `modulation=1.0` (Triangle modulation)

which addresses a moderate effect strength.

ABv3_CU18_1ch_AOT2_orchestra_mono is used as input for this scene. The output is a 2-channel signal.

9.3.4.6.2.2 Testing of audioCompressor

Abv3_aFXP_aCo01 This scene is used for testing the **audioCompressor** effect. At start-up the effect is disabled (bypass of the clip). At t=6s the parameters inside the **audioFXParams** field are set to:

`nfloor: -90dB; threshold`: -90dB
`loknee`, `hiknee`: -50dB
`ratio` 10:1
`att, rel` (=`attack, release`): 25ms (uncritical value)
`look, delay`: not used

which is a strong compression setting to hear the effect.

ABv3_CU18_1ch_AOT2_orchestra_mono is used as input for this scene. The output is a 1-channel signal.

9.3.4.6.2.3 Testing of audioEcho

ABv3_aFXP_aEc01 This scene is used for testing the **audioEcho** effect. At start-up the effect is disabled (bypass of the clip). At t=6s the parameters inside the **audioFXParams** field are set to:

`Channel`: 1.0
`taps`: 1.0
`feedback`: 25.0 (=25%)
`delay time`: 440.0 (=440 ms)

This is a simple 1 tap echo after 440ms that should be heard clearly.

ABv3_CU18_1ch_AOT2_orchestra_mono is used as input for this scene. The output is a 1-channel signal.

9.3.4.6.2.4 Testing of audioEqualizer

Abv3_aFXP_aEq01 This scene is used for testing the **audioEqualizer** effect. At start-up the effect is set to a shelving filter with 0dB gain.

`numFreqBands`: 2.0
`centerFreq`: 200.0 3000.0
`bandwidth`: 800.0 2000.0
`gain`: 1.0 1.0

At t=3s the bass and treble gain are set to -6dB, at t=6s they are set to 0dB and at t=9s they are set to +6dB. These changes should be noticed clearly.

ABv3_CU17_1ch_AOT2_orchestra is used as input for this scene. The output has 2 channels.

Abv3_aFXP_aEq02 This scene is used for testing the **audioEqualizer** effect. At start-up the effect is set to a 9 channel octave band equalizer with 0dB gain.

`numFreqBands`: 9.0
`centerFreq`: 63.0 125.0 250.0 500.0 1000.0 2000.0 4000.0 8000.0 16000.0
`bandwidth`: 45.0 88.0 177.0 354.0 707.0 1414.0 2828.0 5657.0 11314.0
`gain`: 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0

At t=3s, t=6s and at t=9s the gains are changed. These changes should be noticed clearly.

ABv3_CU17_1ch_AOT2_orchestra is used as input for this scene. The output has 2 channels.

9.3.4.6.2.5 Testing of audioFilter

Abv3_aFXP_aFi01 This scene is used for testing the **audioFilter** effect. At start-up the effect is disabled (bypass of the clip):

`filtertype`: 0.0
`filterorder`: 0.0
`frequency`: 2000.0
`bandwidth`: 0.0
`characteristic`: 1.0

At t=6s the parameters inside the **audioFXParams** field are set to a Bessel lowpass 4th order at 2000Hz by changing `filterorder` to 4.0.

ABv3_CU17_1ch_AOT2_orchestra is used as input for this scene. The output has 2 channels.

9.3.4.6.2.6 Testing of audioFlange

Abv3_aFXP_aFl01 This scene is used for testing the **audioFlange** effect. At start-up the effect is disabled (bypass of the clip). At t=6s the parameters inside the **audioFXParams** field are set to:

`rate`: 2.0
`depth`: 0.5
`modulation`: 1.0
`feedback`: 0.7

which is a medium effect strength setting.

ABv3_CU18_1ch_AOT2_orchestra_mono is used as input for this scene. The output has 2 channels.

9.3.4.6.2.7 Testing of audioNaturalReverb

Abv3_aFXP_aNa01 This scene is used for testing the **audioNaturalReverb** effect. After start-up a 40564 samples impulse response will be loaded by sending 317 consecutive blocks of the **audioFXParams** field from t=4 ms until t=1268 ms. The indication of the desired load sequence will be done by setting the start-up set of the **audioFXParams** to

`numParamsFields`: 317.0
`numImpResp`: 1.0
`sampleRate`: 44100.0
`reverbChannels`: 1.0
`impulseResponseCoding`: 0.0 (=consecutive samples)

The transmitted impulse response should be compared with the original impulse response in Abv3_AudioNaturalReverb_ImpulseResponse.wav.

ABv3_CU18_1ch_AOT2_orchestra_mono is used as input for this scene. The output is a 1 channel signal.

9.3.4.6.2.8 Testing of audioReverb

Abv3_aFXP_aRe01 This scene is used for testing the **audioReverb** effect. At start-up the effect is disabled (bypass of the clip). At t=6s the preset 'church' will be selected by setting the parameters inside the **audioFXParams** field to:

`preset`: 4.0
`length`: 0.0
`numFreqBands`: 0.0
`frequencyBand`: 0.0
`reverberation`: 0.0

ABv3_CU18_1ch_AOT2_orchestra_mono is used as input for this scene. The output is a 1 channel signal.

9.3.4.6.2.9 Testing of audioSpeedChange

This scene is used for testing the **audioSpeedChange** effect. At start-up the effect is disabled (bypass of the clip). At t=6s the parameter inside the **audioFXParams** field is set to:

`speedFactor`: 0.5

which means half speed.

ABv3_CU18_1ch_AOT2_orchestra_mono is used as input for this scene. The output is a 1 channel signal.