INTERNATIONAL STANDARD

ISO/IEC 14496-4

Second edition
2004-12-15

Information technology — Coding of audio-visual objects —

Part 4: Conformance testing

Technologies de l'information — Codage des objets audiovisuels —

Partie 4: Essai de conformité

Reference number
ISO/IEC 14496-4:2004(E)

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

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.

ISO/IEC 14496-4 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

This second edition cancels and replaces the first edition (ISO/IEC 14496-4:2000), which has been technically revised.

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*

Introduction

Parts 1, 2 and 3 of ISO/IEC 14496 specify a multiplex structure and coded representations of audio-visual information. Parts 1, 2 and 3 of ISO/IEC 14496 allow 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, picture size, picture shape, picture rate, bitrate parameters, synchronisation timestamps, the association of bitstreams and synthetic objects within objects, the association of objects within scenes, the protection of bitstreams, objects and scenes. Part 6 of ISO/IEC 14496 specifies a framework for uniform delivery of MPEG-4 content according to the requested associated QoS, irrespective of their location and the transport technology.

This part of ISO/IEC 14496 specifies how tests can be designed to verify whether bitstreams and decoders meet the requirements as specified in parts 1, 2, 3 and 6 of ISO/IEC 14496 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 parts 1, 2 and 3 of ISO/IEC 14496.
- manufacturers of decoders and their customers can use the tests to verify whether the decoder meets the requirements specified in parts 1, 2 and 3 of ISO/IEC 14496 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 parts 1, 2 and 3 of ISO/IEC 14496.

Information technology — Coding of audio-visual objects —
Part 4:
Conformance testing

1 Scope

This part of ISO/IEC 14496 specifies how tests can be designed to verify whether bitstreams and decoders meet requirements specified in parts 1, 2 and 3 of ISO/IEC IEC 14496 and for part 6 of ISO/IEC 14496 it specifies how tests can be designed for bitstream delivery over various delivery technologies in an interoperable transparent manner to parts 1, 2 and 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 bitstreams compliant with the syntactic and semantic bitstream requirements specified in parts 1, 2 and 3 of ISO/IEC 14496.

Characteristics of coded bitstreams and decoders are defined for parts 1, 2 and 3 of ISO/IEC 14496. The characteristics of a bitstream define the subset of the standard that is exploited in the bitstream. Examples are the applied values or range of the picture size 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 coded bitstreams the decoder can decode and reconstruct, by defining the subset of the standard that may be exploited in decodable bitstreams. A bitstream can be decoded by a decoder if the characteristics of the coded bitstream are within the subset of the standard specified by the decoder capabilities.

Procedures are described for testing conformance of bitstreams and decoders to the requirements defined in parts 1, 2 and 3 of ISO/IEC 14496. Given the set of characteristics claimed, the requirements that must be met are fully determined by parts 1, 2 and 3 of ISO/IEC 14496. 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 bitstream and decoder conformance. This document gives guidelines on how to construct bitstream test suites to check or verify decoder conformance. In addition, some test bitstreams implemented according to those guidelines are provided as an electronic annex to this document. The procedures and signaling messages for session and channel establishment are defined in part 6 of ISO/IEC 14496.

Conformance with the signaling messages and procedures in this part of ISO/IEC 14496 are defined in accordance to the specifications in part 6 of ISO/IEC 14496. This specification allows the manufacturer to identify the conformance of the signaling message in a static review and provides abstract test cases to test the conformance to the procedures in a dynamic review of an implementation as defined in ISO/IEC 9646 Conformance Testing standard.

2 Normative references

The following standards contain provisions which, through reference in this text, constitute provisions of this part of ISO/IEC 14496. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this part of ISO/IEC 14496 are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below. Members of IEC and ISO maintain registers of currently valid International Standards.

ISO 639:1988, *Code for the representation of names of languages*

ISO 8859-1, *Information technology — 8-bit single-byte coded graphic character sets — Part 1: Latin alphabet No. 1*

IEC 461:1986, *Time and control code for video tape recorders*

IEC 908:198, *Compact disk digital audio system*

ITU-T Rec. T.81 (1992)|ISO/IEC 10918-1:1994, *Information technology — Digital compression and coding of continuous-tone still images: Requirements and guidelines*

ISO/IEC 9646-1:1994, *Information technology — Open Systems Interconnection — Conformance testing methodology and framework — Part 1: General concepts*

ISO/IEC 9646-2:1994, *Information technology — Open Systems Interconnection — Conformance testing methodology and framework — Part 2: Abstract Test Suite Specification*

ISO/IEC 9646-7:1995, *Information technology — Open Systems Interconnection — Conformance testing methodology and framework — Part 7: Implementation Conformance Statements*

ISO/IEC 11172-1:1993, *Information technology — Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s — Part 1: Systems*

ISO/IEC 11172-2:1993, *Information technology — Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s — Part 2: Video*

ISO/IEC 11172-3:1993, *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 11172-4:1995, *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*

ITU-T Rec. H.222.0(2000)|ISO/IEC 13818-1:2000, *Information technology — Generic coding of moving pictures and associated audio information: Systems*

ITU-T Rec. H.262(1995)|ISO/IEC 13818-2:1996, *Information technology — Generic coding of moving pictures and associated audio information: Video*

ISO/IEC 13818-3:1998, *Information technology — Generic coding of moving pictures and associated audio information — Part 3: Audio*

ISO/IEC 13818-7:1997, *Information technology — Generic coding of moving pictures and associated audio information — Part 7: Advanced Audio Coding (AAC)*

ISO/IEC 14496-1:2001, *Information technology — Coding of audio-visual objects — Part 1: Systems*

ISO/IEC 14496-2:2001, *Information technology — Coding of audio-visual objects — Part 2: Visual*

ISO/IEC 14496-3:2001, *Information technology — Coding of audio-visual objects — Part 3: Audio*

ISO/IEC 14496-6:2000, *Information technology — Coding of audio-visual objects — Part 6: Delivery Multimedia Integration Framework (DMIF)*

Recommendations and reports of the CCIR, 1990, XVIIth Plenary Assembly, Dusseldorf, 1990 Volume XI — Part 1: Broadcasting Service (Television) ITU-R BT.601-5, *Studio encoding parameters of digital television for standard 4:3 and wide-screen 16:9 aspect ratios*

CCIR Volume X and XI Part 3 Rec. 648: *Recording of audio signals*

CCIR Volume X and XI Part 3 Report 955-2: *Sound broadcasting by satellite for portable and mobile receivers, including Annex IV Summary description of advanced digital system II*

IEEE Standard Specifications for the Implementations of 8 by 8 Inverse Discrete Cosine Transform, IEEE Std 1180-1990, December 6, 1990

ITU-T Rec. H.261, *Codec for audiovisual services at px64 kbit/s,* Geneva, 1990

3 Terms and definitions

Relevant definitions for this part of ISO/EC 14496 can be found in ISO/IEC 14496-1, ISO/IEC 14496-2, ISO/IEC 14496-3 and ISO/IEC 14496-6 for Systems, Visual, Audio and DMIF definitions respectively.

Relevant abbreviations and symbols for this part of ISO/EC 14496 can be found in ISO/IEC 14496-1, ISO/IEC 14496-2, ISO/IEC 14496-3 and ISO/IEC 14496-6 for Systems, Visual, Audio and DMIF definitions respectively.

4 Systems

4.1 Conformance Points

Figure 1 illustrates a typical MPEG-4 terminal, as per the specifications of the Systems Decoder Model as identified in ISO/IEC 14496-1. With reference to this model, the following conformance point types have been identified.

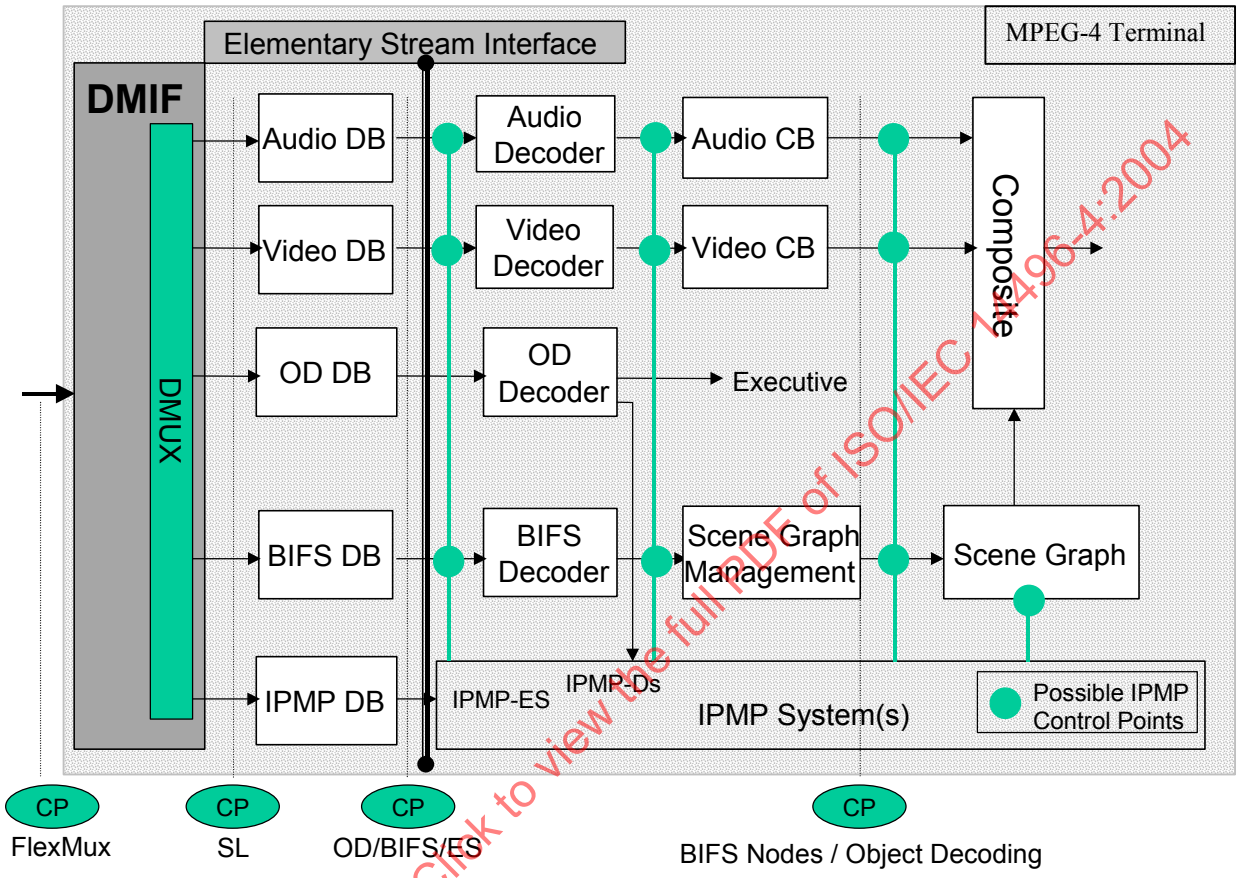


Figure 1 — Typical MPEG-4 terminal

Figure 1, DB are Decoding Buffers, CB are Composition Buffers. Audio CB contain PCM data. Video CB contain pixel data. Decoding buffers contain reconstructed Access Units (AU) or pieces of AU.

Bitstream conformance points are:

- FlexMux
- Synchronisation Layer
- OD Decoding
- BIFS Decoding
- OCI Decoding
- IPMP
- Systems Decoder Model conformance

At a bitstream conformance point, bitstreams will be acquired for use in testing.

Terminal conformance points are:

- FlexMux
- Synchronisation Layer
- OD Decoding Buffer
- BIFS Decoding Buffer

- OCI Decoding Buffer
- IPMP
- Scene Graph
- Systems Decoder Model conformance

4.1.1 FlexMux Conformance Point

A FlexMux conformance point is a conformance point where FlexMux streams as defined in subclause 12.2 of ISO/IEC 14496-1 can be acquired or inserted. According to a scene delivery, there may be several FlexMux conformance points. Each FlexMux conformance points correspond to one FlexMux channel allocated under DMIF responsibility. A FlexMux conformance point can be envisaged according to a bitstream point of view and according to a Terminal point of view. FlexMux bitstream conformance points are dedicated to the syntactic aspect of the FlexMux streams that can be acquired, while FlexMux Terminal conformance points are more dedicated to the semantics and the coherence of the FlexMux-ed streams, which can be acquired or inserted, with their associated signalling. The MPEG-4 signalling can be found in the Object descriptors.

4.1.2 Sync Layer Conformance Point

A Synchronisation Layer (SL) conformance point has to be considered from two possible points of view : the SL bitstream point of view and the SL Terminal point of view. SL bitstream conformance points are dedicated to the syntactic aspect of the SL bitstreams which can be acquired or inserted, assuming that the SL configuration of each SL stream is known upon acquisition of the Object Descriptor. SL terminal conformance points are more dedicated to the semantics and the coherence of the SL bitstreams with the associated signalling acquired from the Object descriptors, with the information found in the related SLConfigDescriptor, and with the information found in the associated SL_PDU packet headers.

4.1.3 OD Conformance Point

This is a point situated between the DMIF interface and the OD parser/decoder. Access Units from OD Elementary Streams are present at this point in the terminal.

4.1.4 BIFS Conformance Point

This is a point situated between the DMIF interface and the BIFS parser/decoder. Access Units from BIFS Elementary Streams are present at this point in the terminal. BIFS Elementary Streams contains BIFS Command Frames or BIFS Anim Frames.

4.1.5 OCI Conformance Point

This is a point situated between the DMIF interface and the OCI parser/decoder. Access Units from OCI Elementary Streams are present at this point in the terminal.

4.1.6 IPMP Conformance Point

IPMP information shall be conveyed in an MPEG-4 bitstream using the IPMP framework described in ISO/IEC 14496-1, subclauses 8.3 and 8.8. This includes the IPMP Elementary stream (IPMP-ES) and the IPMP Descriptors (IPMP-Ds). IP Identification information shall be conveyed using IPI Data sets as specified in ISO/IEC 14496-1, subclause 8.6.8. IPMP bitstream conformance points are dedicated to syntactic conformance. IPMP terminal conformance points are dedicated to semantic conformance.

4.1.7 Scene Graph Conformance Point

This is a point situated between the Scene Graph Management and the Compositor. The data present at this point represents the current state of the Scene Graph, i.e. the integration over time of all BIFS Commands and BIFS Anims received by the terminal as well as all interactions from the viewer.

It is the last point in the BIFS information flow where conformance can be specified. The format of the data at this point is implementation-dependent. However, there shall be a way to extract this implementation-dependent information and present it for conformance testing in the Scene Dump format specified in the Test Material subclause below.

4.2 Bitstream Conformance

Each bitstream shall meet the syntactic and semantic requirements specified in ISO/IEC 14496-1. This subclause describes a set of tests to be performed on bitstreams. In the description of the tests it is assumed that the tested bitstream 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. Note that the application of these tests requires parsing of the bitstream to the appropriate levels. Parsing and interpretation of ODs is also required. In some cases of IPMP-protected data, de-scrambling may be required before the tests can be performed on non IPMP-related features.

4.2.1 FlexMux Conformance

4.2.1.1 Conformance Requirements

FlexMux-ed bitstreams shall comply with the specifications in subclause 12.2 of ISO/IEC 14496-1.

4.2.1.2 Measurement procedure

Syntax of the bitstream shall meet the requirements of subclause 12.2 of ISO/IEC 14496-1.

4.2.1.3 Tolerance

There is no tolerance for bitstream syntax checking. The diagnosis is pass or fail.

4.2.2 Synchronization Layer Conformance

4.2.2.1 Conformance Requirements

SL-packetized bitstreams shall comply with the specifications in subclause 10.2 of ISO/IEC 14496-1.

4.2.2.2 Measurement procedure

Syntax of the SL Packets shall meet the requirements of subclause 10.2 of ISO/IEC 14496-1.

4.2.2.3 Tolerance

There is no tolerance for bitstream syntax checking. The diagnosis is pass or fail.

4.2.3 OD Conformance

4.2.3.1 Conformance Requirements

OD streams shall comply with the specifications in clause 8 of ISO/IEC 14496-1.

4.2.3.2 Measurement procedure

Syntax of the OD stream shall meet the requirements of clause 8 of ISO/IEC 14496-1.

4.2.3.3 Tolerance

There is no tolerance for bitstream syntax checking. The diagnosis is pass or fail.

4.2.4 BIFS Conformance

4.2.4.1 Conformance Requirements

BIFS streams shall comply with the specifications in subclause 9.3 of ISO/IEC 14496-1.

4.2.4.2 Measurement procedure

Syntax of the BIFS stream shall meet the requirements of subclause 9.3 of ISO/IEC 14496-1.

4.2.4.3 Tolerance

There is no tolerance for bitstream syntax checking. The diagnosis is pass or fail.

4.2.5 OCI Conformance

4.2.5.1 Conformance Requirements

OCI descriptors included in ObjectDescriptors or ES_Descriptors shall comply with ISO/IEC 14496-1 subclause 8.4. A conformant OCI bitstream shall only contain OCI events and OCI descriptors that are compliant to ISO/IEC 14496-1 subclause 8.4. A conformant OCI bit stream shall be embedded in SL bitstreams, the configuration of which complies to ISO/IEC 14496-1 subclause 8.4.2

4.2.5.2 Measurement procedure

Syntax of the OCI stream and of the OCI descriptors shall meet the requirements of subclauses 8.4 and 8.6 of ISO/IEC 14496-1.

4.2.5.3 Tolerance

There is no tolerance. The diagnosis is pass or fail.

4.2.6 IPMP Conformance

4.2.6.1 Conformance Requirements

The IPMP information in a conformant bit stream shall consist only of IPMP-ESs and IPMP-Ds that are compliant to ISO/IEC 14496-1 subclauses 8.3 and 8.8 as well as IPI Data Sets that are compliant to ISO/IEC 14496-1 subclause 8.6.9.

4.2.6.2 Measurement procedure

The IPMP information in a conformant bit stream shall consist only of IPMP-ESs and IPMP-Ds that are parse-able to the extent of the specification of ISO/IEC 14496-1 subclauses 8.3 and 8.8 as well as IPI Data Sets that are parse-able.

4.2.6.3 Tolerance

There is no tolerance for bitstream syntax checking. The diagnosis is pass or fail.

4.2.7 Miscellaneous Conformance

4.2.7.1 Conformance Requirements

4.2.7.1.1 Private data handling

The normal operation of compliant MPEG decoders shall not be affected by the presence of private data in MPEG4 system streams, i.e. decoders shall operate in the same way, if any private data are inserted or are not inserted in the already predefined fields.

Decoders shall be at a minimum capable of parsing and ignoring all private fields.

Decoders shall be at a minimum capable of parsing and ignoring all private elementary streams.

4.2.7.1.2 Buffer management

The SDM testing, in terms of buffer underflow and overflow in the SDM is done one elementary stream at a time.

From a System Decoder Model point of view, FlexMux bitstream compliance, SL and Elementary stream compliance are required.

4.2.7.2 Measurement procedure

All the implied bitstream syntaxes shall meet their associated requirements defined in ISO/IEC 14496-1, clause 7.

4.2.7.3 Tolerance

There is no tolerance. The diagnosis is pass or fail.

4.3 Terminal Conformance

This subclause describes procedures to verify conformance of terminals. Each compliant decoder shall be able to decode all compliant ISO/IEC 14496-1 streams within the subset of the standard defined by the specified capabilities of the decoder.

All tests are performed using error free bitstreams. To test for correct interpretation of syntax and semantics, test sequences covering a wide range of parameters shall be supplied to the decoder under test and its output sequence shall be compared with the known expected output as described for the specific test sequence or bitstream. The comparison can be done, for example, by performing subjective evaluation, by verification of the expected result, or by comparing the timing performance. Such tests are necessary but not sufficient to prove conformance. They are helpful for discovering non-compliant implementations.

Tests are expected to be used for testing ISO/IEC 14496 decoders, including video and audio decoding, as it is generally not practical to test system decoders (or ISO/IEC 14496-1 decoders) alone. Practical test results depend on successful (or expected) output of the entire ISO/IEC 14496 decoder (systems, video, audio and DMIF).

Visual composition conformance is out of the scope of this document, as there is no specification of the visual result of object composition in ISO/IEC 14496-1.

Transport conformance is also out of the scope of this document.

4.3.1 FlexMux conformance

4.3.1.1 Conformance Requirements

The FlexDemux shall recover the SL Packets in the appropriate Decoding Buffer bit-exact as presented to the multiplex, and this for every Elementary Stream present in the FlexMux-ed stream under test.

A maximum bitrate can be specified for each Elementary Stream, see ISO/IEC 14496-1, subclause 8.6.5. Conformant bitstreams shall obey this constraint.

4.3.1.2 Measurement procedure

The recovered SL Packets shall be compared bit-wise with the original packets.

4.3.1.3 Tolerance

There is no tolerance. The diagnosis is pass or fail.

4.3.2 Synchronization Layer Conformance

Although the associated descriptor, called the SLConfigDescriptor, is conveyed as a part of the object descriptor framework, it's conformance issues are of great concern in this subclause since it pertains to the syntax and semantics of the SL-packet headers.

4.3.2.1 Conformance Requirements

The Sync Layer shall recover Access Units (AU) of the embedded Elementary Stream, from the consecutive SL layer packet payload and provide fragments of AU, fragment by fragment , or complete AU, to the associated decoder buffers through the ESI Interface, with the relevant parameters when present in the SL packet headers.

When OCR samples are present, they shall be used to reconstruct the Object Time Base, and shall comply with the timing accuracy conformance described in the following paragraph.

When DTSs and CTSs are present, they shall be coherent with the reconstructed OTB, in order to satisfy the constraint of the System Decoder Model.

On the Sync Layer (ISO/IEC 14496-1, clause 10), the elementary streams are mapped into sequences of SL-packets. The underlying stream that carries these packets is called the SL-Packetized stream (SPS). The Sync Layer specifies a syntax for the packetization of these elementary streams into access units, which are the basic units for time synchronization. The SL-packet consists of a header (SL-packet header) and the payload (SL-packet payload). The header carries the coded representation of time stamps and other associated information necessary for timing and synchronization processes.

This subclause deals with conformance issues related to the sync layer. Although the associated descriptor, called the SLConfigDescriptor, is conveyed as a part of the object descriptor framework, it's conformance issues are included in this subclause since it pertains to the syntax and semantics of the SL-packet headers. The subsequent subclauses deal with the conformance issues related to the SL-packets themselves. It is to be noted that these subclauses are rather incomplete. The Sync layer was designed to be delivery agnostic, i.e., the DMIF provided the interface and exchange between the external delivery layers and the internal elementary stream generation and packetization layers.

NOTE — However, with the ongoing discussions within ISO/IEC JTC 1/SC 29/WG 11 regarding the carriage of MPEG-4 over MPEG-2 transport as well as over IP, the conformance issues regarding the Sync layers must be revisited at the appropriate junctures, in these contexts. However, some of the following will still hold for implementations using the DMIF.

The Sync Layer shall recover the Access Units of the Elementary Stream and store them in the decoder buffer.

4.3.2.1.1 The Synchronization Layer Configuration Descriptor

The descriptor SLConfigDescriptor, which is conveyed within the ES_Descriptor for the elementary stream under consideration, contains the configuration information for the syntax of the SL Packet Headers for the access units in this elementary stream. The syntax of the SLConfigDescriptor is detailed in ISO/IEC 14496-1, subclause 10.2.3. This subclause deals with the syntactic conformance of the SLConfigDescriptor elements.

4.3.2.1.2 Structure

The SLConfigDescriptor element shall have the tag value equal to 0x06.

If predefined = 0x01, the packet header is empty.

If predefined = 0x00:

- If the `useAccessUnitStartFlag` and `useAccessUnitEndFlag` are set to 0, then each Access Unit in the stream is confined to one single SL-packet.
- If `OCRlength` is not 0, then `OCRstreamFlag` shall be set to 0.
- If `OCRstreamFlag`=1, then `OCRlength` shall be set to 0.
- If `OCRstreamFlag`=1, then `OCR_ES_ID` shall be one of the `ES_IDs` of the elementary stream in the same name scope as this elementary stream.

4.3.2.2 Measurement Procedure

4.3.2.2.1 Timing Accuracy (OCR) Procedure

The following paragraph does not replace in any way what is normatively stated in ISO/IEC 14496-1.

The general assumption is that when an OCR sample is included, it refers to the beginning of the byte containing the first bit of the OCR field in the SL_PDU header.

4.3.2.2.1.1 From the transmission point of view

Assumptions:

a) the network provides a constant delay transmission for any bytes of the bitstreams.

b) bitstreams are delivered at a constant bitrate.

c) It is also assumed that constant bitrate means that there is a number 'r' such that for every time interval ΔT the following inequality is satisfied:

$$r * \Delta T - k \leq the\ number\ of\ received\ bytes \leq r * \Delta T + k$$

where k is a constant to capture the divergence from the ideal that we produce when discrete phenomena are modelled through continuous processes.

d) It is assumed that an 'ideal' clock exists, which is approximated by the original Object Time Base.

The OCR values differ from the Object Time Base because of sampling errors while the Object Time Base differs from the ideal time because of differences in nominal and actual clock frequencies.

To test for a constant bitrate we consider all pairs of OCR values, OCR[i], OCR[j], and for every such interval, we consider the ideal and the Object Time Base values as shown below:

Table 1 — Naming Convention for OCRs

byte index	Ideal time	Object Time Base	OCRs
i	T_i	t_i	OCR[i]
j	T_j	t_j	OCR[j]

And as a result we have:

$$OCR[i] = t_i \pm e$$

$$OCR[j] = t_j \pm \delta$$

$$t_j - t_i = [T_j - T_i] * \left[1 + \frac{\Delta f}{f}\right]$$

Where δ is the error on OCR samples, Δf is the mean frequency drift during (T_i, T_j) and f is the nominal clock frequency.

Therefore, we can deduce:

$$(OCR[j] - OCR[i]) - 2 * \delta \leq (t_j - t_i) \leq (OCR[j] - OCR[i]) + 2 * \delta$$

$$\Delta T_m = \frac{(t_j - t_i) - 2 * \delta}{1 + \frac{(\Delta f)_{\max}}{f}} \le T_j - T_i \le \Delta T_M = \frac{(t_j - t_i) + 2 * \delta}{1 - \frac{(\Delta f)_{\max}}{f}}$$

in view of our constant bitrate definition this can be translated into:

$$\frac{\#\ of\ bytes - k}{\Delta T_M} \le r \le \frac{\#\ of\ bytes + k}{\Delta T_m}$$

In the unlikely event where $\Delta Tm \le 0$, the rightmost expression will be treated as $+\infty$.

Since this inequality must hold for all i and all $j \neq i$, we compute:

$$r_{\min} = MAX\left\{\frac{(\#\ of\ received\ bytes - k)}{\Delta T_M}\right\}$$

and

$$r_{\max} = MIN\left\{\frac{(\#\ of\ received\ bytes + k)}{\Delta T_m}\right\}$$

with the minimum and maximum taken for all i and j.

Unless:

$$r_{\min} \le r_{\max}$$

the conformance test failed.

If as an example we take an Object Time Base clock with identical characteristics as the ISO/IEC 13818-1 transport STC, and with OCR samples having the same constraints as the PCR samples:

1. 27MHz as STC frequency,
2. 810 Hz (30 ppm) as STC frequency error,
3. +/- 500 nsec as the allowed tolerance on the PCR samples.

Under these conditions the values for e , Δf_{max} ,f are, respectively, 500 nsec, 810 Hz, and 27MHz.

4.3.2.2.1.2 From a stored bitstream point of view

For a constant bitrate SL stream, the position of the OCR in the bitstream and the value of the OCR are proportional. If no rounding errors are present, the following rule would be obeyed:

$$\frac{OCR(i) - OCR(i')}{(i - i')} = const$$

Where i and i' ($i \neq i'$) are the position indices of the byte in the bitstream containing the first bit of the objectClockReference field and OCR(i) and OCR(i') are the values of the OCR timestamps taken into account the wrap-arounds (see subclause 10.2.7 of ISO/IEC 14496-1).

As each OCR is an integer number, derived from the OTB at the time, the sending terminal generates the OCR time stamp, there may be a rounding error of up to $\pm\delta$. The rounding error of two OCRs may thus accumulate to $\pm 2\delta$. The exact value of the constant will thus be in the interval given by:

$$\left[\frac{|OCR(i) - OCR(i')| - 2\delta}{|i - i'|}, \frac{|OCR(i) - OCR(i')| + 2\delta}{|i - i'|}\right]$$

Thus there must exist one value of *const* such that the following inequality holds for all values of i and i' to which an OCR can be attached:

$$const - \frac{2\delta}{|i-i'|} \le \frac{OCR(i) - OCR(i')}{(i-i')} \le const + \frac{2\delta}{|i-i'|}$$

The value of const has to be calculated in high precision. In practical cases, the size of the bitstream is finite, which means, that the value of *const* can only be determined to be within some interval [const $_{min}$, const $_{max}$].

Since these two inequalities must hold for all i and all i' ≠ i, we compute:

$$const_{\min} = MAX\left\{-\frac{2\delta}{|i-i'|} + \frac{OCR(i) - OCR(i')}{(i-i')}\right\}$$

and

$$const_{\max} = MIN\left\{\frac{2\delta}{|i-i'|} + \frac{OCR(i) - OCR(i')}{(i-i')}\right\}$$

with the minimum and maximum taken for all i and i'.

Unless:

$$const_{\min} \le const_{\max}$$

the conformance test failed.

NOTE 1 — As we deal with the position of the OCR in the bitstream, there is no tolerance in the frequency error. The result of a frequency shift is a time varying delivery rate which can not be checked here.

NOTE 2 — If the value of *const* has rounding errors, these rounding errors also will have to be taken into account for the definition of the above interval.

NOTE 3 — This should be rediscussed should δ be restricted to 0.5

4.3.2.2.2 Timestamping (DTS & CTS) Procedure

For a constant bitrate SL stream containing OCR information, decoding and composition time stamps shall be tested the following way:

1. verify that the OCR test has been successfully passed
2. decode a time stamp (decoding or composition) taking into account the wrap-arounds (see subclause 10.2.7 of ISO/IEC 14496-1)
3. scan the bitstream to end of AU
4. calculate the OTB of the next byte, taking into account the wrap-arounds.

The DTS and CTS time stamps values shall obey the SDM constraints and shall be greater or equal to the OTB determined in 4.

4.3.2.3 Tolerance

There is no tolerance. The diagnosis is pass or fail.

4.3.3 OD Conformance

4.3.3.1 Conformance Requirements

Within an object descriptor stream, new object descriptors shall be encapsulated within an ObjectDescriptorUpdate command.

Each access unit, corresponding to an object descriptor stream, shall contain the object descriptor commands in their entirety, i.e., an `ObjectDescriptorUpdate` or an `ObjectDescriptorRemove` command shall not go over one access unit.

All the commands encapsulated within one access unit shall have the same time stamp and shall be processed at the same instant of time, corresponding to the values of the time stamps in the SL Header.

The ObjectDescriptorUpdate command shall have its tag value equal to 0x01.

The ObjectDescriptorRemove command shall have its tag value equal to 0x02.

4.3.3.1.1 Structure for URLs

This subclause briefly discusses the structure of the URL string as it will be used in the remote invocation of string and services. The actual URL protocols and structures are out of scope of the ISO/IEC 14496-1 specifications. However, the bitstream representation of these strings must be compliant with the ISO/IEC 10646:2000 and its amendments (or the Unicode 2.0 and its amendments) specifications. If the URLs in the Object Description Framework are specified to have a certain structure, then these may be included in the conformance specifications in the future drafts.

4.3.3.1.2 The Initial Object Descriptor

This subclause looks into the conformance requirements for an InitialObjectDescriptor. The syntax and semantics for this descriptor are detailed in ISO/IEC 14496-1, subclause 8.6.3.

The structural conformance is a part of the syntactic conformance.

4.3.3.1.2.1 Structure

Shall have its tag value equal to 0x02.

Shall have an ObjectDescriptorID value not equal to 0x000.

If the URL_Flag is set to 0, the InitialObjectDescriptor shall indicate the following:

- ODProfileLevelIndication
- sceneProfileLevelIndication
- audioProfileLevelIndication
- visualProfileLevelIndication
- graphicsProfileLevelIndication

If the URL_Flag is set to 0, the InitialObjectDescriptor shall also aggregate at least one ES_Descriptor element.

The InitialObjectDescriptor may aggregate at most 30 ES_Descriptor elements.

An InitialObjectDescriptor may aggregate up to a maximum of 255 OCI_Descriptors.

An InitialObjectDescriptor may aggregate up to a maximum of 255 IPMP_DescriptorPointers.

An InitialObjectDescriptor may aggregate additional descriptors, called ExtensionDescriptors, but up to a maximum of 255 in number (see ISO/IEC 14496-1, subclause 8.6.15).

If the URL_Flag is set to 1, the URL string shall be a ISO/IEC 10646:2000 (or Unicode 2.1) compliant string.

4.3.3.1.2.2 Scope of URLs in the Initial Object Descriptor

Shall point to a location whose content shall be an InitialObjectDescriptor.

4.3.3.1.3 The Object Descriptor

4.3.3.1.3.1 Structure

An object descriptor shall be encapsulated within an ObjectDescriptorUpdate command.

Shall have the tag value equal to 0x01.

Shall have a unique 10-bit ObjectDescriptorID within the name scope not equal to 0x000.

Bits 11-15 within an object descriptor shall be set to 1. This count does not include the bits for indicating the tag values.

An object descriptor shall aggregate of only ES_Descriptors, OCI Descriptors and IPMP descriptor pointers, in that order.

If the URL_Flag = 0, the object descriptor shall aggregate at least one ES_Descriptor. The aggregation of ES_Descriptors of various streamType values is described below.

An object descriptor may aggregate up to a maximum of 30 ES_Descriptors.

An object descriptor may aggregate up to a maximum of 255 OCI_Descriptors.

An object descriptor may aggregate up to a maximum of 255 IPMP_DescriptorPointers.

Independently of the URL_Flag, an object descriptor may aggregate ExtensionDescriptors, up to a maximum of 255 in number (see ISO/IEC 14496-1, subclause 8.6.15).

4.3.3.1.3.2 Aggregation of ES_Descriptors in an Object Descriptor

This subclause pertains to the cases wherein a given object descriptor aggregates more than one ES_Descriptor elements.

All specifications and restrictions detailed in ISO/IEC 14496-1, subclause 8.7.1, shall be fulfilled.

4.3.3.1.4 Scope of URLs in Object Descriptors

URLs in object descriptors shall point to object descriptor elements at local or remote locations. The stream received from the remote location shall be a ObjectDescriptorUpdate command encapsulating a new object descriptor.

4.3.3.1.5 Elementary Stream Descriptors

This subclause deals with the conformance specifications as related to the ES_Descriptors (ISO/IEC 14496-1, subclause 8.6.4). The first subclause delves into the syntactic conformance of the ES_Descriptor element. The subsequent subclause delve into the dependencies of elementary streams on each other.

4.3.3.1.5.1 Structure

ES_Descriptors shall be encapsulated within a new Object Descriptor when making a reference to a new audio-visual object.

If updating the ES_Descriptors for an existing Object Descriptor, the new ES_Descriptors shall be encapsulated within ES_DescriptorUpdate commands and shall refer to this existing object descriptor.

To change the attributes of an elementary stream as conveyed by an ES_Descriptor, it is required that the existing ES_Descriptor associated with this elementary stream shall be removed (via the ES_DescriptorRemove command) and the new ES_Descriptor shall be conveyed (encapsulated in the ES_DescriptorUpdate command). The conveyance of this new ES_Descriptor shall follow the rules outlined in (1) and (2) in this subclause.

An ES_Descriptor element shall have its tag value equal to 0x03.

The ES_Descriptor element shall have a unique 16-bit ES_ID.

If the value of streamDependenceFlag is set, the 16-bit dependsOn_ES_ID of this ES_Descriptor element shall have the value of the ES_ID of one of the other ES_Descriptor elements aggregated in the same object descriptor. The streamDependenceFlag of the latter ES_Descriptor element shall be 0 and the streamTypes of the two ES_Descriptor elements shall be the same.

An ES_Descriptor shall aggregate one DecoderConfigDescriptor and one SLConfigDescriptor.

An ES_Descriptor shall aggregate at most one IPI_DescrPointer and at most one Qos_Descriptor.

An ES_Descriptor shall aggregate at most 255 IPMP_DescriptorPointer elements and at most 255 LanguageDescriptor elements.

Each ES_Descriptor shall have either one IPI_DescrPointer or (0…255) IP_IdentificationDataSet elements.

Each ES_Descriptor is scoped within the name space of the parent object descriptor. In other words, a given object descriptor is not aware of an ES_Descriptor element that it did not aggregate.

An ES_Descriptor aggregates un number of Descriptors such as DecoderConfigDescriptor , SLConfigDescriptor, IPI_DescrPointer, Qos_Descriptor. IPMP_DescriptorPointer, LanguageDescriptor elements, IPI_DescrPointer, IP_IdentificationDataSet , which shall appear in the order and shall obey the rules defined in ISO/IEC 14496-1, subclause 8.6.4.

4.3.3.1.5.2 Elementary Stream Dependencies

This subclause delves a bit further into the dependencies between elementary streams.

All specifications and restrictions detailed in ISO/IEC 14496-1, subclause 8.7.1.5, shall be fulfilled.

4.3.3.1.5.3 Scope of URLs in ES_Descriptors

The URLs in ES_Descriptors shall point to elementary streams. It is expected that the `streamType` of the ES_Descriptor and the stream type of the referred elementary stream are the same.

4.3.3.1.5.4 Name Scope of Identifiers

The scope of the `ObjectDescriptorID`, `ES_ID` and `IPMP_DescriptorID` identifiers that label the object descriptors, elementary stream descriptors and IPMP descriptors, respectively, is defined as follows. This definition is based on the restriction that associated scene description and object descriptor streams shall always be aggregated in a single object descriptor, as specified in subclause 8.6.2 of ISO/IEC 14496-1. The following rules define the name scope:

- Two `ObjectDescriptorID`, `ES_ID` or `IPMP_DescriptorID` identifiers belong to the same name scope if and only if these identifiers occur in elementary streams with a `streamType` of either ObjectDescriptorStream or SceneDescriptionStream that are aggregated in a single object descriptor.

4.3.3.1.5.5 Reuse of identifiers

For reasons of error resilience, it is recommended not to reuse `ObjectDescriptorID` and `ES_ID` identifiers to identify more than one object or elementary stream, respectively, within one presentation. That means, if an object descriptor or elementary stream descriptor is removed by means of an OD command and later on reinstalled with another OD command, then it should still point to the same content item as before.

4.3.3.1.6 Decoder Configuration Descriptors

The descriptor DecoderConfigDescriptor (ISO/IEC 14496-1, subclause 8.6.5) provides the information for the configuration of the elementary stream decoders. This subclauses addresses some of the syntactic conformance elements for this descriptor.

4.3.3.1.6.1 Structure

The DecoderConfigDescriptor shall have a tag value of 0x04.

The `objectTypeIndication` value shall not be 0x00.

The `streamType` value shall not be 0x00.

If `streamType` = 0x04, the `objectTypeIndication` attribute shall take on one of the values from 0x20, 0x60-0x65, 0x6A and 0xFF. The last value shall indicate that no profile is specified.

If `streamType` = 0.x05, the `objectTypeIndication` attribute shall take on one of the values from 0x40, 0x66-0x69, 0x6B and 0xFF. The last value shall indicate that no profile is specified.

4.3.3.2 Tolerance

There is no tolerance. The diagnosis is pass or fail.

4.3.4 BIFS Conformance

4.3.4.1 Conformance Requirements

The terminal shall recover the BIFS Elementary Stream in the BIFS Decoding Buffer bit-exact as constructed by the BIFS encoder.

4.3.4.2 Measurement Procedure

The BIFS Access Units recovered from this conformance point shall be strictly identical to the Access Units stored in the corresponding BIFS track in the test MP4 file.

4.3.4.3 Tolerance

There is no tolerance. The diagnosis is pass or fail.

4.3.5 OCI Conformance

4.3.5.1 Conformance Requirements

The OCI decoder shall produce or modify the list of events associated to an Elementary stream, in concordance with the OCI events contents.

The OCI decoder shall monitor the incoming events associated to an Elementary stream, in concordance with their associated timing.

The classification Entity defined within the Content Classification descriptor shall be one value provided by the registration authority to the organisation who provided the Content Classification Descriptor.

The rating Entity defined within the Rating descriptor shall be one value provided by the registration authority to the organisation who provided the Rating Descriptor.

4.3.5.2 Measurement procedure

This procedure is application dependant.

4.3.5.3 Tolerance

The tolerance is application dependant.

4.3.6 IPMP Conformance

4.3.6.1 Conformance Requirements

A conformant ISO/IEC 14496 terminal shall pass all IPMP-ESs and IPMP-Ds to the appropriate IPMP System as indicated by the *IPMP_Type* of ISO/IEC 14496-1 subclauses 8.3.2 and 8.6.13, if an according IPMP System is available in the terminal.

4.3.6.2 Measurement Procedure

An ISO/IEC 14496-1 conformant terminal shall be able to parse the IPMP descriptors and IPMP ES (subclause 8.3.2 of ISO/IEC 14496-1 and subclause 8.6 of ISO/IEC 14496-1) and the IPI data sets (subclause 8.6.8 of ISO/IEC 14496-1) to the extent of ISO/IEC 14496-1.

4.3.6.3 Tolerance

There is no tolerance. The diagnosis is pass or fail.

4.3.7 Scene Graph Conformance

4.3.7.1 Conformance Requirements

The scene shall be reconstructed and updated by BIFS-Command streams, BIFS-Anim streams and ROUTEs execution as specified in ISO/IEC 14496-1, subclause 9.3.

4.3.7.2 Measurement procedure

The scene graph shall be the same as the scene graph of a reference implementation at any time. The procedure to test is to make scene dumps according to the format described in this document at some key points in time. The test material must provide the original BIFS bitstream as well as the scene dumps for the same key points in time. The key points will be determined by the author of the test sequence according to the following criteria:

- in the case of BIFS-Anim or BIFS-Command streams, the scene graph shall be checked after the CTS of each command or anim value.
- in the case of interpolators activated by ROUTEs, the scene graph shall be checked every 100 ms. The assumption is that interpolation of values are performed at the same rate shifted by 50 ms.

4.3.7.3 Tolerance

The accuracy of the time stamp at which the values will be updated shall be tested according to the level definition.

Arithmetic precision shall be tested according to the level definition.

4.3.8 Miscellaneous Conformance

4.3.8.1 Conformance Requirements

On every conformance point to be tested, the acquired bitstreams shall be compliant with the related bitstream conformance tests, and the insertion of compliant bitstreams shall not induce incoherent particular and general behavior in the terminal process.

4.3.8.1.1 BIFS acquisition

Terminals shall use the InitialObjectDescriptor, the BIFS Command, ObjectDescriptor and IPMP information to support acquisition of any MPEG-4 scene.

As during the duration of a scene, the scene definition will change, the IPMP, InitialObjectDescriptor, the BIFS Command and ObjectDescriptor information have to be continuously monitored.

4.3.8.1.2 Handling of discontinuities

In compliant MPEG-4 systems streams, not at every Access Unit boundary, but on some particular Access Unit, discontinuities in any decoding process can occur (visual decoding, audio decoding, System's bitstream decoding).

Assuming that any combination of changes in decoding process parameters which lead to parameter values that are supported by the decoder under test, the terminal under test shall:

- Maintain correct composition synchronisation between the different Elementary streams
- Not produce unacceptable audio or visual artefacts

4.3.8.1.3 Private data handling

The normal operation of compliant MPEG decoders shall not be affected by the presence of private data in MPEG-4 system streams, i.e. decoders shall operate in the same way, if any private data are inserted or are not inserted in the already predefined fields.

Decoders shall be at a minimum capable of recognising and ignoring all private fields.

Decoders shall be at a minimum capable of recognising and ignoring all private elementary streams.

4.3.8.2 Measurement Procedure

4.3.8.2.1 Buffer overflow/underflow tests

Continuous OTB shall be present and available.

The SDM buffer fullness will be continuously monitored with the use of CTS and DTS timestamps (when present), and with the use of the OTB. When present, continuous DTS and CTS shall be available for such conformance test.

4.4 Test material and test suites

This subclause contains the description of test material and test suites required by the previous subclauses.

The test sequences are packaged in MP4 file format. The MP4 file format specification can be found in the ISO/IEC 14496-1 working draft for Amendment 1.

The test sequences are bundled with an HTML file describing:

- the title of the sequence,
- the authors,
- the reference to the clause(s) of this document that this test sequence pertains to,
- the content, at the level of detail needed to be able to perform the test,
- the list and nature of the other documents.

Some test sequences may also be accompanied by:

- parsing hints, helping the tester to locate errors by using textual comparison of the parsing hints with a log of the parsing by the decoder under test,

- scene dumps, allowing the comparison of the actual scene tree in the tested decoder with the scene tree as specified by the standard.

The textual file formats to be used in the other documents are described in the next two subclauses.

4.4.1 Parsing Hint File Format

4.4.1.1 Requirements

The log files are to fulfil the following requirements to facilitate conformance testing:

- easily legible and understandable for human beings
- easy automatic comparison, e.g. by the UNIX command “diff”

4.4.1.2 Syntax Elements in a Log File

It is suggested that any line in a log file should correspond to exactly one single read (for decoders) or write (for encoders) operation of an ISO/IEC 14496 syntax element (see subclause 4.4.1.2.1 for details). The order of lines in a log file shall correspond to the order of the decoding process as is given by the syntax descriptions in the relevant parts of ISO/IEC 14496.

Any such line in a log file shall contain the following syntax elements (the angled brackets are to be skipped in a real log file, whereas the round ones are to be kept):

<TOS> (<bits>, <Tbits>) <semantic> <bits read> <blank> <decoded value>

Table 2 explains the meaning of the different syntax elements in one log file line. For easier legibility as well as for automatic processing it also describes:

- the exact *starting position* counted in characters from the beginning of the line (e.g. <TOS> always starts with the 1st character in a line)
- the *field width* in characters for the syntax elements (e.g. for <bits> there shall always be 3 characters reserved); Those parts of the fields that are not needed actually are to be left blank
- the *alignment* of the syntax elements (e.g. the numbers for <bits> and <Tbits> will be easier legible being right-aligned)

Table 2 — Interpretation of syntax elements in a log file

Syntax element	Meaning	starting position	field width	Alignment
<TOS>	Indicates the type of the stream from which the bits are read acc. to Table 3	1	9	left
(	Separator for easier human legibility	10	1	-
<bits>	number of bits used to encode the semantic	11	3	right
,	Separator for easier manual legibility	14	1	-
<Tbits>	number of bits read altogether so far (since start of decoding process)	15	10	right
)	Separator for easier manual legibility	25	4	left
<semantic>	the textual description of the bits read, according to the syntax used in ISO/IEC 14496, see also subclause 4.4.1.2.1	29	32	left
<bits read>	the bits read, interpreted as a hexadecimal number	61	17	right
<blank>	blank characters for better legibility	78	2	-
<decoded value>	see subclause 4.4.1.2.1	80	N/A	left

Table 3 indicates the strings that are to be used for <TOS>.

This value is to be followed by the stream’s ES_ID as a hexadecimal number separated from the first part by one blank character.

Table 3 — Values for <TOS>

stream type	<TOS>
InitialObjectDescriptor	**IOD**
ObjectDescriptorStream	**OD** *ES_ID*
SceneDescriptionStream	**BIFS** *ES_ID*
ObjectContentInfoStream	**OCI** *ES_ID*
ClockReferenceStream	**OCR** *ES_ID*
IPMPStream	**IPMP** *ES_ID*
AudioStream	**AUD** *ES_ID*
VisualStream	**VID** *ES_ID*

4.4.1.2.1 <semantic> syntax element

As stated in Table 2, the <semantic> field shall provide for the textual description of the bits read, according to the syntax used in ISO/IEC 14496. I.e., every sophisticated ISO/IEC 14496 syntax element that is being constructed from other syntax element has to be broken down recursively to primitive syntax elements that cannot be broken down any further.

E.g., there would be no(!) <semantic> value *Transform2D*. Instead, every node would have to be broken down by its fields. The fields in turn would have to be broken down further until the level of definition where bit(), int(), float() or double() appear is reached.

4.4.1.2.2 <decoded value> syntax element

The <decoded value> syntax element shall reflect the interpretation of the bits read, according to the value of the <semantic> element. E.g. if the value read from the bitstream is of type SFBoolean the <decoded value> element would be equal to either *TRUE* or *FALSE* depending on the actual value in the bitstream. In another example the <semantic> element might indicate that the bits are to be interpreted as a *nodeType*. Hence <decoded value> would simply be equal to the name of the node (e.g. *Transform2D* or *Bitmap* or ...).

However, although the above examples are rather straight-forward the definition of the different possible values of the <decoded value> syntax element requires a lot of work due to the high amount of data types that are permitted (e.g. see subclause 9.3.7 of ISO/IEC 14496-1).

Therefore, this field shall be left open for additional but non-normative information, which should provide some useful information for human readers[1]. Guidelines for the provision of useful information by this syntax element are given in the following subclause.

4.4.1.2.2.1 Guidelines for useful information in the <decoded value> syntax element

The <decoded value> syntax element shall contain information about the bits that have been decoded, in the form that makes sense for them. For example boolean values shall be written as TRUE or FALSE. In cases where the bits represent an enumerated type such as *nodeType*, a textual value of the enumerated type shall be printed. In cases where no information is needed as in the case of an integer, the field may be left blank. String values shall be printed as is. Any other comments can be added to this field as is felt necessary.

4.4.1.3 Example

The following line are to serve as an example of a line arbitrarily chosen from a log file (note that here the <semantic> field width is reduced to 30 character for better legibility and that I also left out the <decoded value> syntax element):

```
BIFS 5    (  1,          3)   isReused                        0  FALSE
```

This line corresponds to the following information:

- one bit has been taken from a scene description stream
- the stream's ES_ID is 5

---

1 Note that for automatic reading and comparison with other files skipping of this syntax element is very easy, since it is located at the end of every line starting at the 78th character.

- the bit read is the third bit taken from this stream so far
- the bit will be interpreted as an *isReused* syntax element, probably inside an SFNode (further information on this will be provided by the context which in turn would be given by the preceding lines in the log file!)
- the bit's (hexadecimal) value is zero
- since *isReused* is an SFBoolean value, its value "0" is interpreted as "FALSE"

4.4.1.4 Suffix for Log Files

For easy recognition the name of every log file shall be terminated by the suffix ".log" leading to the format ***.log** for any such file's name.

4.4.2 Scene Dump File Format

The interchange format is an XML text file. The file contains a description of all nodes, routes, and fields of the current state of the scene. The structure of this file is intended to simplify the parsing and identification of various parts of the scene graph.

Parsers must skip over any elements and attributes that are not defined in this subclause.

4.4.2.1 Elements and their attributes

4.4.2.1.1 <scene-dump>

This element brackets the data to be interchanged. It is the top-level element of the file.

Container:
The file.

Attributes:
version-number Required. Set to "1.0"

4.4.2.1.2 <timestamp>

Contains terminal's session time value (at the point the scene dump is captured) expressed in SFTime format.

Container:
<scene-dump>

Attributes:

reference-value Required if the scene is timed to a clock reference stream. Contains a snapshot of the clock reference as an integer. Note that this value will wrap and cannot be used as the sole indicator of session time.

4.4.2.1.3 <scene>

This element is a container for all nodes of the scene graph.

Container:
<scene-dump>

Attributes:
(none)

4.4.2.1.4 <node>

This element describes a node in the scene graph. It contains all the fields of the node. If the node is a reference to an already instanced node, the field dump is optional.

Container:
<scene>, <indexed-value>, <node>

Attributes:

type Required. Contains the integer SFWorld node encoding type as defined in ISO/IEC 14496-1 Annex H

instance-id Optional. Set to the node ID for instanced nodes.

use-id Optional. Set in nodes that are reusing an instanced node.

name Optional. Contains the name of the node.

4.4.2.1.5 <field>

This element describes a field in a node. All fields of type field or exposedfield are dumped. Scalar field values are written in the same text form as defined in ISO/IEC 14772-1.

Container:
<node>

Attributes:

def-id Required. Contains the defID of the field as defined in ISO/IEC 14496-1 Annex H.

name Optional. Contains the name of the field.

4.4.2.1.6 <indexed-value>

This element is used to contain a single value in an MF-type field.

Container:
<field>

Attributes:
index Required. Contains the zero-based integer index of the value in the field.

4.4.2.1.7 <routes>

This element serves as a container for all the route definitions in the scene.

Container:
<scene-dump>

Attributes:
(none)

4.4.2.1.8 <route>

This element describes a route in the scene.

Container:
<routes>

Attributes:

id Required. Contains the route's id.

src-node Required. Contains the instance identifier of the route's source node.

src-field Required. Contains the outID of the source field within the source node.

dst-node Required. Contains the instance identifier of the route's target node.

dst-field Required. Contains the inID of the target field of the route's target node.

4.4.2.2 Example file

```
<?xml version="1.0"?>
<scene-dump version="1">
        <timestamp reference-value=="1234">0.0</timestamp>
        <scene>
                <node name="Group" type="46" instance-id="1">
                        <field name="children" def-id="0">
                                <indexed-value index="0">
                                        <node name="Transform" type="94" instance-id="2">
                                                <field name="translation" def-id="5">0.0 0.0 0.0</field>
                                                <field    name="rotation"    def-id="2">0.0    1.0    0.0
0.0</field>
                                        </node>
```

```
                                </indexed-value>
                                <indexed-value index="1">
                                        <node name="TimeSensor" type="92" instance-id="3">
                                                <field name="cycleInterval" def-id="0">10000</field>
                                                <field name="enabled" def-id="1">TRUE</field>
                                                <field name="loop" def-id="2">TRUE</field>
                                                <field name="startTime" def-id="3">0.0</field>
                                                <field name="stopTime" def-id="4">0.0</field>
                                </indexed-value>
                                <indexed-value index="2">
                                        <node name="OrientationInterpolator" type="66" instance-id="4">
                                                <field name="key" def-id="0">
                                                        <indexed-value index="0">0.0</indexed-value>
                                                        <indexed-value index="1">1.0</indexed-value>
                                                </field>
                                                <field name="keyValue" def-id="1">
                                                        <indexed-value index="0">0.577 0.577 0.577 0.0
                                                        </indexed-value>
                                                        <indexed-value   index="1">0.577   0.577   0.577
6.283185
                                                        </indexed-value>
                                                </field>
                                        </node>
                                </indexed-value>
                        </field>
                </node>
        </scene>
        <routes>
                <route id="1" src-node="3" src-field="6" dst-node="4" dst-field="0"/>
                <route id="2" src-node="4" src-field="2" dst-node="2" dst-field="4"/>
        </routes>
</scene-dump>
```


4.4.3 Test Suites

This paragraph describes the test suites to be used. A test suite is a suite of material and measurement algorithms and associated reference algorithms.

4.4.3.1 BIFS Feature List

The test suite shall verify the features in Table 4. For nodes, the following shall be tested:

- Presence in the scene tree after decoding.
- Appropriate value of the fields after decoding.
- Functionality that has an effect on the scene tree, e.g. for a ROUTE, if the source field value changes, the target field value shall change accordingly.

All of the above shall be checkable through scene dumps as specified in subclause 4.4.2. Rendering not being normative, the aspect of the node is not subject to conformance testing.

Table 4 — BIFS Test Suite Information

N°	Feature	Reference of Test sequence and associated method
1.	BIFS-Anim: position 3D animation	anim-rect, anim1, anim-box1, anim-box2
2.	BIFS-Anim: position 2D animation	anim-simple, anim-rect, anim-circle, anim2

N°	Feature	Reference of Test sequence and associated method
3.	BIFS-Anim: color animation	anim-box2, anim-box1, anim-box, anim1
4.	BIFS-Anim: angle animation	anim-circle, anim-rect
5.	BIFS-Anim: float animation	anim1, anim2
6.	BIFS-Anim: bound float animation	anim1, anim2
7.	BIFS-Anim: normal animation	anim1, anim-box1
8.	BIFS-Anim: size 3D animation	anim-box, anim1, anim-box2
9.	BIFS-Anim: size 2D animation	anim-simple, anim2
10.	BIFS-Anim: integer animation	anim2 (There are no native nodes that have integer animatable fields. This example uses a PROTO. It's the only way to do integer animation and so only one example is provided for this feature).
11.	BIFS-Anim: several fields in the same node	anim-rect, anim-box
12.	BIFS-Anim: several nodes	anim-simple, anim-box
13.	BIFS-Anim: skip frame	No test provided. skipFrame is available for compatibility with FBA, but it is not used in BIFS-anim.
14.	BIFS-Anim: switch of a node (isActive mask)	anim1, anim2
15.	BIFS-Anim: random access true	anim-box1, anim-box2 (any animation other than ANIM 5/anim1 or ANIM 6/anim2)
16.	BIFS-Anim: random access false	Anim1, anim2
17.	Quantization: 3D position	QuantPos3D-4bit, QuantPos3D
18.	Quantization: 2D position	QuantHead2D, QuantPos2D
19.	Quantization: drawing order	QuantDefUse, QuantDefUse1, QuantDrawOrder, QuantQPtest
20.	Quantization: color	QuantColor, QuantQPtest
21.	Quantization: texture coordinate	QuantHead2D, QuantTextureCoord
22.	Quantization: angle	QuantAngle-8bit, QuantAngle, QuantTextureCoord
23.	Quantization: scale	QuantHead2D, QuantQPtest
24.	Quantization: interpolator keys	QuantHead2D, QuantKey
25.	Quantization: normals	Normal-4bit, QuantAngle, QuantQPtest
26.	Quantization: rotations	QuantRotation, QuantQPtest, QuantNormal-4bit
27.	Quantization: object size 3D	QuantObject3D, QuantQPtest
28.	Quantization: object size 2D	QuantObject2D, QuantQPtest
29.	Quantization: linear scalar quantization	QuantQPtest, QuantObject2D
30.	Quantization: efficient float	QuantQPtest, QuantRotation
31.	Quantization: node default values	*Any streams from 17 to 33*
32.	Quantization: isLocal mode	QuantPos2D, QuantPos3D
33.	Quantization: DEF/USE	QuantDefUse, QuantDefUse1
34.	BIFS Command: insert node index	allupdates
35.	BIFS Command: insert node begin	allupdates
36.	BIFS Command: insert node end	Updatetest, Friday, allupdates

N°	Feature	Reference of Test sequence and associated method
37.	BIFS Command: insert Idx value index	allupdates
38.	BIFS Command: insert Idx value begin	allupdates
39.	BIFS Command: insert Idx value end	allupdates
40.	BIFS Command: insert ROUTE	allupdates, slides2
41.	BIFS Command: delete node	Bifs-deletenode
42.	BIFS Command: delete Idx value index	Friday, allupdates
43.	BIFS Command: delete Idx value begin	allupdates
44.	BIFS Command: delete Idx value end	allupdates
45.	BIFS Command: replace node	
46.	BIFS Command: replace field	Bifs-2dfieldreplace1, Friday, allupdates
47.	BIFS Command: replace Idx value index	Pae_raise, allupdates
48.	BIFS Command: replace Idx value begin	allupdates
49.	BIFS Command: replace Idx value end	allupdates
50.	BIFS Command: replace ROUTE	
51.	BIFS Command: replace scene	Ecran2, Updatetest
52.	BIFS Command: several commands in same AU	updatetest
53.	BIFS Scene: mask node	QuantAngle, anim-box1
54.	BIFS Scene: list node	Jerusalem, Layout, Testlayout
55.	BIFS Scene: mask MFField	QuantHead2D, QuantDrawOrder
56.	BIFS Scene: list MFField	QuantHead2D, QuantDrawOrder
57.	BIFS Scene: ROUTE	Scaling3D, Jerusalem, Ecran2
58.	SFBool	Ecran2, Updatetest
59.	SFColor	Ecran2, Updatetest
60.	SFFloat	Ecran2, Updatetest
61.	SFInt32	Ecran2, Updatetest
62.	SFRotation	Normal-4bit, QuantObject3D, jColor3D
63.	SFString	Ecran2, Updatetest
64.	SFTime	Jerusalem, OrientInterp3D
65.	SFUrl	Anchor, Audiotest
66.	SFVec2f	Ecran2, Updatetest
67.	SFVec3f	Bifs-deletenode
68.	SFImage	sfimage-1, sfimage-2
69.	SFCommandBuffer	Ecran2
70.	SFScript	Scaling3D, SFColor01, Value_changed3d, Qtvr
71.	BIFSConfig: BIFS Anim	Anim-rect, Anim-circle, Anim-simple
72.	BIFSConfig: BIFS Command	Ecran2, Jerusalem
73.	Anchor	Anchor, Frame1
74.	AnimationStream	Anim-rect, Anim-circle, Anim-simple
75.	Appearance	Bifs-deletenode, Bifs-2dfieldreplace1
76.	AudioBuffer	
77.	AudioClip	

N°	Feature	Reference of Test sequence and associated method
78.	AudioDelay	
79.	AudioFX	
80.	AudioMix	
81.	AudioSource	Audiotest, Ifs
82.	AudioSwitch	
83.	Background	nist-enst/Bindable_Nodes/Background/*
84.	Background2D	
85.	Billboard	nist-enst/Grouping_Nodes/Billboard/*
86.	Bitmap	Ecran2, Updatetest, Transition
87.	Box	Bifs-deletenode, nist-enst/Geometry/Box/*
88.	Circle	Bifs-2dfieldreplace1, Ecran2, Simple
89.	Collision	nist-enst/Grouping_Nodes/Collision/*
90.	Color	nist-enst/Geometric_Properties/Color/*
91.	ColorInterpolator	Timestest, Anibut3
92.	CompositeTexture2D	Layout, CompositeTexture2D
93.	CompositeTexture3D	
94.	Conditional	Ecran2, Layout, Friday
95.	Cone	Bifs-deletenode, nist-enst/Geometry/Cone/*
96.	Coordinate	nist-enst/Geometric_Properties/Coordinate/*
97.	Coordinate2D	Layout, Updatetest
98.	CoordinateInterpolator	
99.	CoordinateInterpolator2D	IBMCoordinateInterpolator2Ds
100.	Curve2D	Layout, Polygontest
101.	Cylinder	Bifs-deletenode, nist-enst/Geometry/Cylinder/*
102.	CylinderSensor	
103.	DirectionalLight	PointLightPrimitive1-3D
104.	DiscSensor	DiscSensorComplex
105.	ElevationGrid	nist-enst/Geometry/ElevationGrid/*
106.	Expression	
107.	Extrusion	nist-enst/Geometry/Extrusion/*
108.	Face	MinMaxFAP, Marco15, wow25, marco30, op3vis, expressions, emotions, opossum, allfaps, FAPMaxBitrate16, AllFaps_rlint, VisExp, NoFAPMinMax, FAPDCT, FrameDCT
109.	Face (Multiple)	FourFaces
110.	FaceDefMesh	*Any streams from 108 to 115*
111.	FaceDefTables	*Any streams from 108 to 115*
112.	FaceDefTransform	*Any streams from 108 to 115*
113.	FAP	*Any stream from 108*
114.	FDP	*Any stream from 108*
115.	FIT	FIT
116.	Fog	Scaling3D, nist-enst/Bindable_Nodes/Fog/*

N°	Feature	Reference of Test sequence and associated method
117.	FontStyle	Scaling3D, Ecran2
118.	Form	Form_spread, Form_spread2, Testform
119.	Group	Anchor, Ecran2, Layout
120.	ImageTexture	Ecran2, Jerusalem, Pae_raise
121.	IndexedFaceSet	Test2, Test3
122.	IndexedFaceSet2D	Ifs
123.	IndexedLineSet	
124.	IndexedLineSet2D	Polygontest, Updatetest, Mosaic18
125.	Inline	
126.	LOD	
127.	Layer2D	Bifs-2dfieldreplace1, Transition
128.	Layer3D	Bifs-deletenode, Scaling3D
129.	Layout	Jerusalem, Layout, Testlayout
130.	LineProperties	Ecran2, Updatetest
131.	ListeningPoint	
132.	Material	Bifs-deletenode, Material3D
133.	Material2D	Bifs-2dfieldreplace1, Ecran2
134.	MovieTexture	Jerusalem, Friday, Av
135.	NavigationInfo	nist-enst/Bindable_Nodes/NavigationInfo/*
136.	Normal	nist-enst/Geometric_Properties/Normal/*
137.	NormalInterpolator	
138.	OrderedGroup	Form_spread2, Pae_raise
139.	OrientationInterpolator	OrientInterp3D
140.	PixelTexture	
141.	PlaneSensor	
142.	PlaneSensor2D	Slider, Valuator
143.	PointLight	PointLightPrimitive1-3D
144.	PointSet	
145.	PointSet2D	PointSet2D, MovingPointFish2D
146.	PositionInterpolator	Value_changed3d
147.	PositionInterpolator2D	Friday, Traj0
148.	ProximitySensor2D	ProxSensInterp2D, ProximitysensorSimple2D
149.	ProximitySensor	
150.	QuantizationParameter	QuantQPtest, QuantDefUse
151.	Rectangle	Ecran2, Updatetest, Friday
152.	ScalarInterpolator	Trans-group
153.	Script	Scaling3D, SFColor01, Value_changed3d, Qtvr
154.	Shape	Bifs-deletenode, Ecran2, Jerusalem
155.	Sound	
156.	Sound2D	Audiotest, Ifs
157.	Sphere	Bifs-InsertNodeStress
158.	SphereSensor	

N°	Feature	Reference of Test sequence and associated method
159.	SpotLight	PointLightPrimitive1-3D
160.	Switch	Ecran2, Jerusalem, Friday
161.	TermCap	
162.	Text	Ecran2, Jerusalem, Updatetest
163.	TextureCoordinate	
164.	TextureTransform	nist-enst/Geometric_Properties/TextureCoordinate/*
165.	TimeSensor	OrientInterp3D, Jerusalem, Trans-group, Timestest
166.	TouchSensor	Scaling3D, Ecran2, Jerusalem, Friday
167.	Transform	Bifs-deletenode
168.	Transform2D	Bifs-2dfieldreplace1, Ecran2
169.	Valuator	Slider, Valuator
170.	Viewpoint	Scaling3D, nist-enst/Bindable_Nodes/Viewpoint/*
171.	VisibilitySensor	
172.	Viseme	
173.	WorldInfo	
174.	DEF / USE	SFColor01, Ecran2, Jerusalem

4.4.3.2 OD Feature List

Table 5 — OD Test Suite Information

N°	Feature	Reference of Test sequence and associated method
1.	IOD	Anchor, Audiotest, Ecran2
2.	OD Update (new)	Ecran2, Jerusalem
3.	OD Remove	
4.	ES Update (new)	
5.	ES Remove	
6.	IPMP Update	
7.	IPMP Remove	
8.	OD Update (modification)	
9.	ES Update (modification)	
10.	OCI descriptors	
11.	IPI descriptors	
12.	QoS descriptors	
13.	Extension descriptors	Ecran2, Slider

4.4.3.3 Bitstreams

Table 6 – Test Sequence Providers and Reason for Existence

Name	Provider	Content
Anchor	ENST	Anchor node
Audiotest	ENST	Audiosource and Sound2d
Ecran2	ENST	Medium size sample
Form_spread	ENST	Form node
Form_spread2	ENST	Form node
Form_spread3	ENST	Form node
Updatetest	ENST	Updates
Transit	ENST	Layer2D as clipping etc…
Valuator	ENST	Valuator
Simple	ENST	Simple2D sample
Jerusalem	ENST	Medium size sample
Layout	ENST	Medium size sample
Pae_raise	ENST	OrderedGroup, updates and interactivity
Polygontest	ENST	Polygons and lines
Slider	ENST	Valuator…
Timestest	ENST	ColorInterpolator
Trans-group	ENST	ScalarInterpolator
Testlayout	ENST	Layout
Testform	ENST	Form
Qtvr	ENST	Script, Valuator, Arb. Shape video
Friday	ENST	Medium size example
Traj	ENST	PositionInterpolator2D
Ifs	ENST	IndexedFaceSet2D
Anim-simple	FT	Animation of Transform2D.scale
Anim-rect	FT	Animation of Transform2D.translation and rotation
Anim-circle	FT	Animation of Transform2D.rotationAngle
Bifs-deletenode	FT	Delete node on 3d nodes
Bifs-2dfieldreplace1	FT	Replace field on 2d nodes
Bifs-InsertNodeStress	FT	Insert node
PointLightPrimitive1-3D	FT	3d lights
OrientInterp3D	FT	Orientation Interpolator
Material3D	FT	Material3D
Angle-8bit	FT	Quantization of angle
Normal-4bit	FT	Quantization of normal
Pos3d-4bit	FT	Quantization of position 3d
Scaling3D	FT	Script
SFColor01	FT	Script
Value_changed3d	FT	Script
BB	Optibase	IndexedFaceSet
Biliard	Optibase	IndexedFaceSet

Name	Provider	Content
Anibut3	ENST	ColorInterpolator, OrderedGroup, ScalarInterpolator, TimeSensor
Av	ENST	Sound2D, MovieTexture
Frame1	ENST	Anchor, etc…
Imabut	ENST	Image button
Interleaved_2s	ENST	Interleaved MP4 file (onechunk is the non-interleaved version)
Kang	ENST	Video with shape (static texture and shape)
Forme	ENST	Video with shape (static shape, moving texture)
Oiseau	ENST	PlaneSensor2D, Video with shape (static texture, moving shape)
Mosaic18	ENST	Background2D, Anchor, etc…
Mosaic41	ENST	Background2D, ScalarInterpolator, PlaneSensor2D…
Slides2	ENST	SlideShow
Facesetgalore	ENST	IndexedFaceSet2D and lots of updates on it.
Allupdates	ENST	Lots of updates encapsulated in Conditionals tied with text buttons.
Interactive	ENST	Tests the interactive starting of media.
Meteo2	ENST	Image and text interactivity.
Paepopup	ENST	A kind of popup menu.
Ultrasimple	ENST	For the simple profile.
Update2	ENST & FT	A set of 17 sequences covering all types of updates
AABphys1-80	HUT	Tests Advanced AudioBIFS physical approach, nodes DirectiveSound, AcousticScene, AcousticMaterial
AABper1-76	FT	Tests Advanced AudioBIFS perceptual approach, nodes DirectiveSound, PerceptualParameters

4.5 Advanced BIFS

4.5.1 Bitstream conformance

4.5.1.1 Conformance Requirements

BIFS streams shall comply with the specifications in clause 9 of ISO/IEC 14496-1.

4.5.1.2 Measurement procedure

Syntax of the BIFS stream shall meet the requirements of clause 9 of ISO/IEC 14496-1.

4.5.1.3 Tolerance

There is no tolerance for bitstream syntax checking. The diagnosis is pass or fail.

4.5.2 Terminal conformance

4.5.2.1 Conformance Requirements

The terminal shall recover the BIFS Elementary Stream in the BIFS Decoding Buffer bit-exact as constructed by the BIFS encoder.

4.5.2.2 Measurement Procedure

The BIFS Access Units recovered from this conformance point shall be strictly identical to the Access Units stored in the corresponding BIFS track in the test MP4 file.

4.5.2.3 Tolerance

There is no tolerance. The diagnosis is pass or fail.

4.6 MPEG-J

4.6.1 MPEG-J Conformance Points

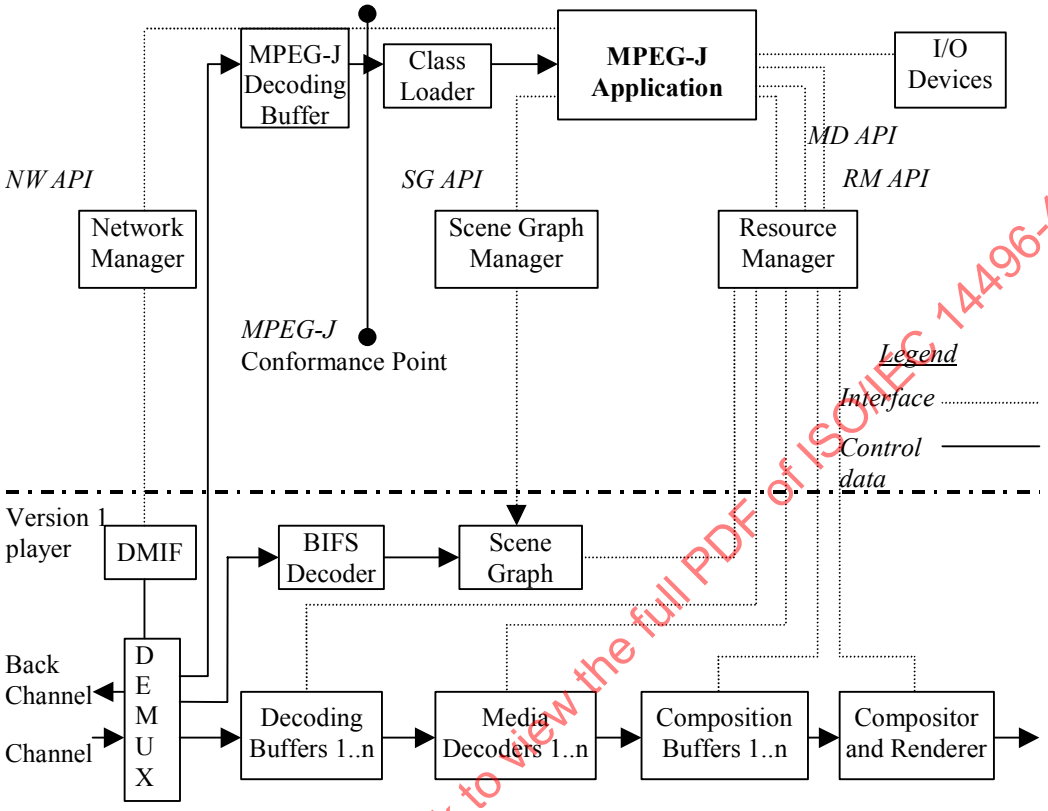


Figure 2 — MPEG-J Architecture with Conformance Point

Architecture of MPEG-J is explained in ISO/IEC 14496-1 subclause 11.2. MPEG-J data is defined and the delivery mechanism explained in ISO/IEC 14496-1 subclause 11.4. MPEG-J data is delivered as an elementary stream similar to video, audio and other elementary streams.

This is de-multiplexed and stored in MPEG-J Decoding Buffers. This buffer feeds the MPEG-J Decoder which "decodes" it. In the case of class (Java byte code), decoding means loading, while for the object and other data it is made available to the terminal.

The MPEG-J Decoding Buffer consists of MPEG-J Access Units defined in subclause. Each MPEG-J Access Unit contains either one class or one serialized object or one archive (a zip file) with a header. When this is decoded, the class file or the object data or the zip file is extracted and fed into the MPEG-J Class Loader as shown in Figure 2.

Bitstream conformance point for MPEG-J is:

- MPEG-J Decoding

At a bitstream conformance point, bitstreams will be acquired for use in testing.

Terminal conformance point for MPEG-J is:

- MPEG-J Decoding Buffer
- MPEG-J API conformance
- Java Platform conformance

An MPEG-J conformance point can be either an MPEG-J bitstream conformance point or an MPEG-J Terminal conformance point. The MPEG-J bitstream conformance points deal with the syntactic aspects while the MPEG-J terminal conformance points address the semantics.

4.6.2 Bitstream Conformance

Each bitstream shall meet the syntactic and semantic requirements specified in ISO/IEC 14496-1. This subclause describes a set of tests to be performed on bitstreams. In the description of the tests it is assumed that the tested bitstream 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. Note that the application of these tests requires parsing of the bitstream to the appropriate levels. Parsing and interpretation of ODs is also required. In some cases of IPMP-protected data, de-scrambling may be required before the tests can be performed on non IPMP-related features.

4.6.2.1 MPEG-J Conformance

4.6.2.1.1 Conformance Requirements

MPEG-J bitstreams shall comply with the specifications in ISO/IEC 14496-1 clause 11. The terminal shall strictly adhere to the syntax specified in 11.4.3.

When the bitstream carries classes, these classes shall only use the classes, interfaces, or API (Application Programming Interface) calls from the following:

1. MPEG-J APIs defined in the ISO/IEC 14496-1 (org.iso.*) for the relevant profile.
2. Java APIs supported by the underlying Java Platform for the relevant profile. These are (typically) in the java.* packages.
3. Classes or Interfaces carried in the bitstream.

These classes shall obey the security rules defined in subclause 11.3.5 of ISO/IEC 14496-1.

4.6.2.1.2 Measurement procedure

Syntax of the bitstream shall meet the requirements of subclause 11.4.3 of ISO/IEC 14496-1.

The classes should compile with only the Java Platform APIs and the MPEG-J APIs relevant to that profile.

Verification mechanism: The API implementations should output a trace file for every bitstream. This trace files should be compared to see if the behavior is the same in two implementations. This idea is similar to the dump format used for BIFS.

`Method packagename.classname.methodName with parameter parameter`$_1$ `parameter`$_2$ `parameter`$_3$`… parameter`$_n$ `was called`

where: *method_name* is the name of the method, parameter$_n$ is:

- value of the parameter - when it is a primitive data type
- the instance name - otherwise.

E.g. a method foo(var$_1$, var$_2$) would print the trace

`Method org.iso.mpeg.mpegj.foo with parameter var`$_1$ `var`$_2$

`Exception packagename.`*`exception_name`* `was thrown ( or )`

`Exception packagename.`*`exception_name`* `was thrown or with parameter var`$_1$

4.6.2.1.3 Tolerance

There is no tolerance for bitstream syntax checking. The diagnosis is pass or fail.

4.6.3 Terminal Conformance

This subclause describes procedures to verify conformance of terminals. Each compliant decoder shall be able to decode all compliant ISO/IEC 14496-1 streams within the subset of the standard defined by the specified capabilities of the decoder.

All tests are performed using error free bitstreams. To test for correct interpretation of syntax and semantics, test sequences covering a wide range of parameters shall be supplied to the decoder under test and its output sequence shall be compared with the known expected output as described for the specific test sequence or

bitstream. The comparison can be done, for example, by performing subjective evaluation, by verification of the expected result, or by comparing the timing performance. Such tests are necessary but not sufficient to prove conformance. They are helpful for discovering non-compliant implementations.

Tests are expected to be used for testing ISO/IEC 14496 decoders, including video and audio decoding, as it is generally not practical to test system decoders (or ISO/IEC 14496-1 decoders) alone. Practical test results depend on successful (or expected) output of the entire ISO/IEC 14496 decoder (systems, video, audio and DMIF).

4.6.3.1 MPEG-J conformance

4.6.3.1.1 Conformance Requirements

Figure 2 shows the architecture an MPEG-J Terminal and the conformance points. The terminal shall follow all the rules regarding:

- MPEG-J Session and Lifecycle specified in subclause 11.3 of ISO/IEC 14496-1.
- MPEG-J Decoding and Loading specified in subclause 11.4 of ISO/IEC 14496-1.
- Semantics of the timestamps specified in sub subclause 11.4.2 of ISO/IEC 14496-1.

All the defined and normatively referred APIs defined subclause 11.5 of ISO/IEC 14496-1 in shall be strictly followed.

4.6.3.1.1.1 MPEG-J Decoding

The Decoding process of MPEG-J data involves two steps:

a. Recovering the access unit data (class, object, or zip file) from the bit stream. This is input to the MPEG-J Class Loader.

b. Loading:

- If the data is a class file it is loaded according to the rules specified in subclause 11.4 of ISO/IEC 14496-1.
- If the data is a zip file the classes specified in the header are loaded according to the rules specified in subclause 11.4 of ISO/IEC 14496-1.
- If the data is neither a class or a zip file, it is made available according to the rules specified in subclause 11.4 of ISO/IEC 14496-1.

4.6.3.1.1.2 MPEG-J API conformance

The terminal shall implement all the APIs that are defined or normatively referenced by ISO/IEC 14496-1 for the relevant profile.

4.6.3.1.1.3 Java Platform conformance

The Terminal shall implement the Java Platform according to the profile. This is further elaborated in Annex A and in the Java Technology Test Suite Development Guide.

4.6.3.1.2 Measurement procedure

The recovered MPEG-J data (classes, objects, or zip files) shall be compared bit-wise with the original data.

The terminal shall strictly adhere to the class/interface definition in subclause 11.5 of ISO/IEC 14496-1. For e.g., the class/interface names, method signatures, variable (if any) names and types, constant names and values shall be as defined.

The measurement procedure for Java Platform conformance is described in detail in Annex E and in the Java Technology Test Suite Development Guide.

4.6.3.1.3 Tolerance

There is no tolerance. The diagnosis is pass or fail.

4.7 MP4 File Format

4.7.1 Writing

If an atom defined in this specification is written, it must be formatted to this specification.

A valid MP4 file with no tracks has at least: moov, mvhd. If it is a presentation or the target of an OD URL, an iods is required, containing an IOD or OD respectively. Only MP4 files used in editing (as the target of a data reference URL) may lack the IOD.

Any track must contain: trak, tkhd, mdia. A mdia must contain mdhd, hdlr, and minf. A minf must contain a suitable media header, a dinf and a stbl. dinf must contain dref; and a stbl must contain stsd, and if there are any samples, an stts, stsz, stco, stsc.

The sample table entries must be consistent about the number of samples in a track.

Extensions should use the UUID mechanism.

Track identifiers must be unique within the file.

The containment hierarchy defined in the specification must be followed. Very few atoms (for example, user data and UUID extension) are allowed to occur in multiple containers.

Fields marked reserved should be written to the standard value.

A presentation must contain at least a BIFS track, referenced by the IOD (as in the systems specification).

4.7.2 Reading

A reader shall be able to scan an atom-formatted file, with any atoms types in it (standard or non-standard). This includes atoms with UUID and length escapes (extended length and indefinite length).

For all atoms within this specification, the structure must be decoded and the correct behavior implemented. Note that there is no normative handling of UUID atoms. Note that the version field of atoms should be checked; unrecognized versions of atoms should be treated as unknown atoms, as a change in version will in general signify a change in structure.

Relative URLs in data references must be accepted; there are no normatively required access methods for absolute URLs, therefore a reader is free to reject MP4 files which use access methods it does not implement.

Other non-standard atoms may be present, but they must not use the types defined here, and it must be possible for a system to deliver the presentation correctly while ignoring them. A compliant reader must skip unrecognized atoms (both those using compact and UUID types).

Fields marked reserved should not be checked on reading; any value should be accepted.

5 Visual

5.1 Introduction

In this clause, except where stated otherwise, the following terms are used for practical purposes:

The term 'bitstream' means ISO/IEC 14496 video bitstream. A bitstream is the coded representation of one layer of a single visual object. A bitstream may contain I-VOPs, P-VOPs, B-VOPs and S-VOPS.

A "visual-object bitstream collection" is a set of bitstreams that represent all the layers of one VO.

A "visual-clip bitstream collection" is a set of bitstreams that represent all the layers of all the visual objects making a video clip.

The term 'decoder' means ISO/IEC 14496 video decoder or ISO/IEC 14496 scalable still texture decoder, i.e., an embodiment of the decoding process specified by ISO/IEC 14496-2. The decoder does not include the display process or composition, which are outside the scope of this standard. The output of a decoder is specified in clause 7 of ISO/IEC 14496-2.

A bitstream is the input to a single elementary stream decoder. This input may not be accessible in a production decoder.

The test output from a decoder is the VO obtained by combining the outputs of the elementary stream decoders for the layers of the VO in accordance with the decoder description of ISO/IEC 14496-2. This VO is extracted from the decoder prior to composition. This output may not be accessible in a production decoder.

The term ‘reference software decoder’ means one of the two software decoders contained in ISO/IEC 14496-5 It is possible to use this software to test and verify that some of the requirements specified in ISO/IEC 14496-2 are met by the bitstream.

If any statement stated in this subclause accidentally contradicts a statement or requirement defined in ISO/IEC 14496-2, the text of ISO/IEC 14496-2 prevails.

The following subclauses specify the normative tests for verifying compliance of video bitstreams, visual-object bitstream collections, visual-clip bitstream collections, video decoders and scalable still texture object decoders. Those normative tests make use of test data (bitstream test suites) provided as an electronic annex to this document, and of a reference software decoder specified in ISO/IEC 14496-5 with source code available in electronic format.

5.2 Definition of visual bitstream compliance

An ISO/IEC 14496 video bitstream is a bitstream that implements the specification defined by the normative clauses of ISO/IEC 14496-2 (including all normative annexes of ISO/IEC 14496-2).

A compliant bitstream, visual-object bitstream collection or visual-clip bitstream collection shall meet all the requirements and implement all the restrictions defined in the generic syntax defined by the ISO/IEC 14496-2 specification, including the restrictions defined in clause 9 of ISO/IEC 14496-2 for the profile-and-level specified the bitstream.

Subclause 5.5 defines the normative tests that a bitstream, visual-object bitstream collection or visual-clip bitstream collection shall pass successfully in order to be claimed compliant with this specification.

A compliant bitstream of a given profile-and-level may be called an “ISO/IEC 14496-2 *Profile@Level* bitstream” or simply a “*Profile@Level* bitstream” (e.g. an MP@L1 bitstream).

5.2.1 Requirements and restrictions related to profile-and-level

The profile_and_level_indication shall be one of the valid codes defined in Annex G of ISO/IEC 14496-2. The profile-and-level derived from the profile_and_level_indication indicates that additional restrictions and constraints have been applied to several syntactic and semantic elements, as defined in ISO/IEC 14496-2 (clause 9, Annex G and Annex N).

The restrictions defined for a given profile-and-level are aimed at reducing the cost of decoder implementation and at facilitating interoperability. A compliant bitstream, visual-object bitstream collection or visual-clip bitstream collection shall be decodable by any compliant ISO/IEC 14496 visual decoder that supports the profile-and-level combination specified in the bitstream.

5.2.2 Additional restrictions on bitstream applied by the encoder

The video encoder or scalable still texture encoder may apply any additional restrictions on the parameters of the video bitstream, in addition to restrictions defined in the generic video syntax and the restrictions defined for the specified profile-and-level in clause 9 of ISO/IEC 14496-2. Not all additional restrictions can be known a priori without analyzing or decoding the entire bitstream, since the syntax does not provide explicit mechanisms which signal such restrictions in advance for all cases.

5.2.3 Encoder requirements and recommendations

5.2.3.1 Encoder requirements

Although encoders are not directly addressed by ISO/IEC 14496-2, an encoder is said to be an ISO/IEC 14496-2 Profile@Level encoder if it satisfies the following requirements:

1) The bitstreams generated by the encoder are compliant Profile@Level bitstreams.

2) For encoding methods which include embedded decoding operations to produce the coded bitstream, these decoding operations shall be performed with the full arithmetic precision specified in ISO/IEC 14496-2.

This second requirement is necessary to guarantee that only compliant decoders will produce images that have optimum quality.

With this requirement on ISO/IEC 14496-2 encoders, any compliant decoder decoding a bitstream generated by a compliant encoder will normally reconstruct images of higher quality, compared to the images reconstructed from the same bitstream by a non-compliant decoder.

5.2.3.2 Encoder recommendations

It is strongly recommended that video encoders capable of producing P-pictures implement the note of subclause 7.4.4 of ISO/IEC 14496-2. Failure to implement this recommendation may cause significant accumulation of mismatch between the reconstructed samples produced by the hypothetical decoder sub-loop

embedded within an encoder and those produced by a (downstream) decoder using the coded bitstream produced by the encoder.

5.2.3.3 Restrictions on the Operation of the Reference Software

The reference software decoder contained in ISO/IEC 14496-5 implements the full elementary stream syntax defined in ISO/IEC 14496-2. For visual_object_type == "video ID", the reference software begins decoding a bitstream at the video_object_start_code. For visual_object_type == "still texture ID", the reference software begins decoding a bitstream beginning with StillTextureObject(). It does not, however, implement the following additional restrictions defined by ISO/IEC 14496-2:

- verification of the constraint imposed on VBV, VCV and VMV, and
- profiles and levels.

In addition, the buffer intercept method defined below is not implemented by this software.

5.3 Procedure for testing bitstream compliance

A bitstream, visual-object bitstream collection or visual-clip bitstream collection that claims compliance with this standard shall pass the following normative test:

When processed by the reference software decoder, the bitstream, visual-object bitstream collection or visual-clip bitstream collection shall not cause any error or non-conformance messages to be reported by the decoder. This test shall be applied only to bitstreams that are known to be free of errors introduced by transmission.

Successfully passing the reference software decoder test only provides a strong presumption that the bitstream under test is compliant, i.e. that it does indeed meet all the requirements specified in ISO/IEC 14496-2 that are tested by the reference software decoder.

Additional tests may be necessary to check more thoroughly that the bitstream implements properly all the requirements specified in ISO/IEC 14496-2. These complementary tests may be performed using other video bitstream verifiers that perform more complete tests than those implemented by the reference software decoder.

ISO/IEC 14496-2 contains several informative recommendations. When testing a bitstream for compliance, it is useful to test whether or not the bitstream follows those recommendations.

To check correctness of a bitstream, it is necessary to parse the entire bitstream and to extract all the syntactic elements and other values derived from those syntactic elements and used by the decoding process specified in ISO/IEC 14496-2 (e.g. vop_height).

A verifier does not necessarily perform all stages of the decoding process described in ISO/IEC 14496-2 in order to verify bitstream correctness. Many tests are performed on syntax elements in a state prior to their use in some processing stages. However, some arithmetic may need to be performed on combinations of syntax elements.

A verifier which does perform the IDCT transform and calculates the reconstructed samples must comply with all the arithmetic precision requirements specified in ISO/IEC 14496-2. In addition, the IDCT of such a verifier shall be an embodiment of the saturated mathematical integer-number IDCT specified in Annex A of ISO/IEC 14496-2 (a software implementation using 64-bit double-precision floating-point is sufficient).

Performing the IDCT and calculating the reconstructed samples in a verifier, although not necessary, is useful for several reasons:

- It allows to test the subjective quality of the reconstructed frames. ISO/IEC 14496-2 does not put any requirement on subjective quality, but it is desirable that an encoder generates bitstreams for which the subjective quality of reconstructed frames is as good as possible.

- Checking the output of the IDCT can provide an indication of whether or not the encoder that produced the bitstream observed the recommendation of the note in subclause 7.4.4 of ISO/IEC 14496-2.

If a bitstream contains a P-VOP with many occurrences of coded blocks of DCT coefficients (i.e., blocks that are not all zeros) for which the output of the reference IDCT is all zeros, then the encoder that produced the bitstream can be suspected of not implementing this important recommendation.

The best chance to discover this problem is when a still image (with no motion at all and no noise) is encoded.

5.4 Definition of visual decoder compliance

In this subclause, except where stated otherwise, the term 'bitstream' means compliant ISO/IEC 14496 visual bitstream (as defined in this part of ISO/IEC 14496) that has the profile_and_level_indication corresponding to the profile-and-level combination considered for the decoder.

Compliance of an ISO/IEC 14496-2 decoder is defined only with regard to a legal profile-and-level combination, as specified in clause 9 of ISO/IEC 14496-2. The decoder shall decode the VOPs of the test bitstreams within the VOP time period indicated in the bitstream (VOP_time_increment). The decoder shall reconstruct I-, P-, B- and S(GMC)-VOPs and sprites within +/-1 pixel difference compared with that generated by the reference software. Additionally the arithmetic accuracy without IDCT of the decoder shall be identical to that of the reference software, except for the warping function of perspective warping used for the decoding of S-VOPs (see subclause 5.4.1). The decoder does not have to display the reconstructed picture within the VOP time period.

The test bitstreams shall stress the decoders by the parameters specified in the profile and level, for example, Max bitrate, MaxObjects, , Max unique Quant Tables, Max VMV occupancy, Max VCV occupancy, Max VBV occupancy, Max video packet length, Max sprite size, Wavelet restrictions, and Combination of tools (e.g, bi-directional prediction with 8x8 MC for all MBs in B-VOP).

NOTE — A compliant decoder may be a special-purpose hardware decoder or a software decoder on a fast enough general-purpose processor dedicated to the operation of the software decoder.

The normative tests that a decoder shall pass in order to claim compliance with a given profile-and-level combination are specified in clause 5.5. A decoder can claim compliance with regard to several profile-and-level combinations if and only if it passes the normative tests defined for each of the profile-and-level combinations.

Only a decoder that passes the conformance test for a given profile-and-level may be called “ISO/IEC 14496-2 *Profile@Level* decoder” or simply “*Profile@Level* decoder” (e.g., an ISO/IEC 14496-2 MP@L2 decoder).

In the following text, decoder compliance is always considered with regard to a particular profile-and-level combination, even when this is not specifically mentioned.

A compliant decoder shall implement a decoding process that is equivalent to the one specified in ISO/IEC 14496-2 and meets all the general requirements defined in ISO/IEC 14496-2 that apply for the profile-and-level combination considered, and if it can decode bitstreams with any options or parameters with values permitted for that profile-and-level combination. The permitted options and parameter range for each profile-and-level combinations are defined in ISO/IEC 14496-2 (clause 9, Annex G and Annex N).

A decoder which implements only a subset of the options or ranges of syntax and semantics for a given profile-and-level combination is not a compliant decoder for that profile-and-level, even if it passes the normative tests specified in clause 5.5. In effect such a decoder would not be capable of decoding all compliant bitstreams of the considered profile-and-level combination.

In the following subclauses the term ‘reference decoder’ means the reference software decoder (ISO/IEC 14496-5).

The reference decoder is a decoder that implements precisely the decoding process as specified in ISO/IEC 14496-2. The IDCT function that shall be used when running the reference decoder is the very accurate approximation of the mathematical saturated integer-number IDCT specified in Annex A of ISO/IEC 14496-2 obtained by implementing IDCT with double-precision arithmetic.

Except for possible mismatches caused by ambiguous half-values rounding at the output of the IDCT and IDWT functions, the output of the reference decoder (reconstructed samples) is defined unambiguously by ISO/IEC 14496-2.

Fundamental requirement areas for decoders are listed in the following subclauses.

5.4.1 Requirement on arithmetic accuracy in video objects (without IDCT)

With the exception of IDCT, the specification of ISO/IEC 14496-2 defines the decoding process absolutely unambiguously. IDCT may yield different results among different implementations. The requirements on the accuracy of the IDCT used by a compliant decoder are specified in Annex A of ISO/IEC 14496-2.

Although unambiguously defined using integer arithmetic, the process of perspective warping (Cf. subclause 7.8.5 of ISO/IEC 14496-2) may require the usage of floating point registers for implementation. The decoder shall calculate the warping functions for perspective warping (i.e. $F(i, j)$, $G(i, j)$, $F_c(i_c, j_c)$, and $G_c(i_c, j_c)$

for the no_of_sprite_point == 4 case defined in subclause 7.8.5 of ISO/IEC 14496-2) within +/-1 difference compared with the values obtained using the integer arithmetic defined in ISO/IEC 14496-2.

There is a requirement that for a block that contains no coefficient data (i.e. if pattern_code[i] is zero, or if the macroblock is skipped) the sample domain coefficients f[x][y] for the block shall all take the value zero (Cf. subclause 7.4.4 of ISO/IEC 14496-2).

Therefore, the following is a the requirement on the arithmetic accuracy of the decoder:

When a coded picture is decoded from a bitstream, for each 8x8 block of the coded picture that is "not-coded" or that contains only zero DCT coefficients, a compliant decoder shall produce reconstructed samples numerically identical to those produced by the reference decoder when the reference frames used by both decoders are numerically identical. A decoder that reconstructs one sample with a value different from that reconstructed by the reference decoder for the same sample is not a compliant decoder.

In other words, all compliant decoders shall produce numerically identical reconstructed samples when the IDCT is applied only to blocks of zero coefficients (assuming that they use numerically identical reference frames).

5.4.2 Requirement on arithmetic accuracy in video objects (with IDCT)

When a bitstream contains some 8x8 blocks with non-zero DCT coefficients, the output of a compliant decoder may differ from the output of the reference decoder. However, because of the accuracy requirements on the IDCT transform used by the decoder, there exist some accuracy requirements on the output of a compliant ISO/IEC 14496 video decoder.

The IDCT used in a compliant decoder shall meet all the requirements defined in Annex A of ISO/IEC 14496-2.

Annex A of ISO/IEC 14496-2 defines additional requirements above those defined by the IEEE Std 1180-1990 standard. In order to claim that the IDCT transform used by the decoder conforms to the specification of Annex A, the IDCT transform shall comply with the IEEE Std 1180-1990 standard and pass successfully the following test:

The test is derived from the specification given in the IEEE Std 1180-1990 standard, with the following modifications:

1) In item (1) of subclause 3.2 of the IEEE specification, the last sentence is replaced by: <<Data sets of 1 000 000 (one million) blocks each should be generated for (L=256, H=255), (L=H=5) and (L=384, H=383). >>

2) The text of subclause 3.3 of the IEEE specification is replaced by : <<For any pixel location, the peak error shall not exceed 2 in magnitude. There is no other accuracy requirement for this test.>>

3) Let F be the set of 4096 blocks Bi[y][x] (i=0..4095) defined as follows :

a) Bi[0][0] = i - 2048

b) Bi[7][7] = 1 if Bi[0][0] is even, Bi[7][7] = 0 if Bi[0][0] is odd

c) All other coefficients Bi[y][x] other than Bi[0][0] and Bi[7][7] are equal to 0

For each block Bi[y][x] that belongs to set F defined above, an IDCT that claims to conform to the specification of Annex A of ISO/IEC 14496-2 shall output a block f[y][x] that as a peak error of 1 or less compared to the reference saturated mathematical integer-number IDCT fíí(x,y). In other words, | f[y][x] - fíí(x,y)| shall be <= 1 for all x and y. Successfully passing the conformance test defined in this document only provides a strong presumption that the IDCT transform is compliant, i.e. that it does meet all the requirements specified in Annex A of ISO/IEC 14496-2. Additional tests may be necessary to check more thoroughly that the IDCT implements properly all the requirements and recommendations specified in Annex A of ISO/IEC 14496-2.

5.4.3 Requirement on arithmetic accuracy in scalable still texture object (without IDWT)

In decoding of scalable still texture object, there is a requirement that if a wavelet transfrom contains no coefficient data , the sample domain coefficients f[x][y] for the frame shall all take the value zero.

Therefore, when a coded image is decoded from a bitstream, if the encoded image only contains only zero DWT coefficients, a compliant decoder shall produce reconstructed samples numerically identical to zero.

5.4.4 Requirement on arithmetic accuracy in scalable still texture (with IDWT)

In decoding of a scalable still texture , when a bitstream contains some nonzero wavelet coefficients, the output of a compliant decoder may differ from the output of the reference decoder. However, because of the accuracy requirements on the IDWT transform used by the decoder, there exist some accuracy requirements on the output of a compliant ISO/IEC 14496 scalable still texture decoder.

The IDWT used in a compliant decoder shall meet all the requirements defined in Annex A of ISO/IEC 14496-2.

In order to claim that the IDWT transform used by the decoder conforms to the specification of Annex A, the IDWT transform shall comply with Annex A.

5.4.5 Requirement on output of the decoding process and timing

The output of the decoding process is specified by subclause 7.13 of ISO/IEC 14496-2.

It is a requirement that all the reconstructed samples of all the coded VOPs be output by a compliant decoder to the display process. For example, a decoder that occasionally does not output some of the reconstructed B-VOPs or that occasionally outputs incomplete reconstructed VOPs to the display process is not compliant. The actual output of the display process is not specified by this standard.

It is a requirement that a compliant decoder outputs the reconstructed samples at the rates specified in subclause 7.13 of ISO/IEC 14496-2.

For example, when decoding an interlaced sequence, there is a requirement that the samples of each field be output to the display process at intervals of 1/(2 * frame_rate).

5.4.6 Recommendations

In addition to the requirements, it is desirable that compliant decoders implement various recommendations defined in ISO/IEC 14496-2.

This subclause lists some of the recommendations.

It is recommended that a compliant decoder be able to resume the decoding process as soon as possible after an error. In most cases it is possible to resume decoding at the next start code or resynchronisation marker.

It is recommended that a compliant decoder be able to perform concealment for the macroblocks or video packets for which all the coded data has not been received.

5.5 Procedure to test decoder compliance

In this subclause, except where stated otherwise, the term 'bitstream' means compliant ISO/IEC 14496 video bitstream (as defined in this document), that has the profile_and_level_indication corresponding to the profile-and-level combination for which conformance of the decoder is considered.

5.5.1 Static tests

Static tests of a video decoder requires testing of the reconstructed samples. This subclause will explain how this test can be accomplished when the reconstructed samples at the output of the decoding process are available.

It may not be possible to perform this type of test with a production decoder. In that case this test should be performed by the manufacturer during the design and development phase.

Static tests are used for testing the arithmetic accuracy used in the decoding process.

There are two sorts of static tests.

- The static tests that do not involve the use of IDCT, IDWT or sprite warping, in which case the test will check that the values of the samples reconstructed by the decoder under test shall be identical to the values of the samples reconstructed by the reference decoder when the reference frames used by both decoders are numerically identical.

- The static tests that involve the use of IDCT, IDWT or sprite warping, in which case the test will check that the peak absolute error between the values of the samples reconstructed by the decoder under test and the values of the samples reconstructed by the reference decoder shall not be larger than 2 when the reference frames used by both decoders are numerically identical.

5.5.2 Dynamic tests

Dynamic tests are applied to check that all the reconstructed samples are output to the display process and that the timing of the output of the decoder's reconstructed samples to the display process conforms to the specification of subclause 7.13 of ISO/IEC 14496-2, and to verify that the decoder buffer verifier models (as defined by Annex D of ISO/IEC 14496-2, VBV, VCV and VMV specification) are not violated when the bits are delivered at the proper rate.

5.5.3 Specification of the test bitstreams

This subclause provides the list of specifications that are used to produce the bitstream test suites for testing decoder compliance. Tests are defined in the following categories:

a) General

b) Shape coding

c) Scalability

d) Error resilience

e) Scalable still texture

f) Sprites

Not all the decoder requirements are covered by these tests, but tests for the most fundamental decoder requirements are believed to be covered by this test suite specification. These tests include :

1. General static tests:

   Bitstreams using all the possible coding options permitted by ISO/IEC 14496-2.

2. Memory bandwidth dynamic tests:

   Bitstreams with all macroblocks predicted with average (bi-directional) prediction, with half-sample interpolation in both the horizontal and vertical directions, for both the luminance and chrominance blocks if possible, using smallest possible prediction blocks and accessing as many different samples of the reference pictures as possible.

3. VLC/FLC decoding static tests:

   Bitstreams using all the possible events within a table.

4. Bits and Symbol distribution (burst) dynamic tests:

   Bitstreams containing very irregular distribution of bits or symbols.

To test a decoder for conformance with regard to a particular profile-and-level combination, a bitstream test suite can be made according to this specification. Each bitstream of the test suite must have its profile_and_level_indication corresponding to the profile-and-level combination considered for the decoder, and must be fully compliant with ISO/IEC 14496-2.

When a bitstream requires the use of an option or parameter value not permitted with the profile-and-level combination considered (e.g., B-VOPs in the case of Simple Profile), the test bitstream must be omitted from the bitstream test suite.

All the bitstreams in the test suite must be such that the output of the non-saturated integer number mathematical IDCT, as defined in Annex A of ISO/IEC 14496-2, has values within the range [-384, 383] for each coded block.

A set of test bitstreams constructed according to selected cases of those specifications is provided in an electronic annex that forms an integral part of this part of 14496. These bitstreams constitute normative test suites that must be used to verify conformance of decoders. The test suites are described in the subclause below.

5.5.3.1 Test Bitstreams – General

In this subclause the number of MB/s allowed is determined by the VCV model as defined in Annex D of ISO/IEC 14496-2.

5.5.3.1.1 Test bitstream #GE-1

Specification: A series of consecutive B-VOPs with all macroblocks using bi-directional interlaced prediction. Number of MB/s and bitrate are the maximum allowed for the profile-and-level combination. Half-sample interpolation in both the horizontal and vertical directions, for all luminance and chrominance blocks.

Functional stage: prediction bandwidth

Purpose: Check that the decoder handles the worst case of prediction bandwidth. Reference VOP buffers organized progressively (interleaved fields) and macroblocks stored in contiguous address page segments would have the greatest penalty. Effective filtered block size is 16x8 for luminance and 8x4 for chrominance.

5.5.3.1.2 Test bitstream #GE-2

Specification: A bitstream with a B-VOP as large as the maximum number of MB/s allowed for the profile-and-level combination, using long VLC's (not via escapes) as much as possible. Number of MB/s and bitrate are the maximum allowed for the profile-and-level combination.

Functional stage: VLD

Purpose: Check that decoder works in this situation. A large B-VOP located after several smaller coded VOPs can catch a decoder off guard.

5.5.3.1.3 Test bitstream #GE-3

Specification: A series of consecutive interlaced coded P-VOPs with all macroblocks using both top and bottom field of the reference VOP. Number of MB/s and bitrate are the maximum allowed for the profile-and-level combination. Maximize number of half-sample prediction in both the horizontal and vertical directions, for both luminance and chrominance blocks.

Functional stage: prediction bandwidth

Purpose: Check that the decoder handles the worst case of prediction bandwidth. Prediction bandwidth is at a maximum in this mode due to the small block sizes and two prediction sources.

5.5.3.1.4 Test bitstream #GE-4

Specification: A bitstream with all macroblock_type transitions progressive and interlaced coded VOPs.

Functional stage: parser

Purpose: Check that decoder handles all scenarios in parsing tree.

5.5.3.1.5 Test bitstream #GE-6

Specification: A bitstream with many different combinations of values for top_field_first, f_codes, quant_type, vop_coded, vop_rounding_type, intra_dc_vlc_thr, alternate_vertical_scan_flag, variable numbers of consecutive coded B-VOPs, coded P-VOPs and coded I-VOPs with downloaded quantization weighting matrices. Ideally the bitstream should contain all possible legal combinations. Various syntax switches are toggled from VOP-to-VOP.

Functional stage: parser and control

Purpose: Check that decoder handle all scenarios.

5.5.3.1.6 Test bitstream #GE-8

Specification: All possible VLC's symbols and IDCT mismatch. Mismatch and saturation.

Functional stage: parser ; IDCT accuracy

Purpose: Test that decoders has included the complete VLC tables and implements mismatch control.

5.5.3.1.7 Test bitstream #GE-9

This test has been removed from the test suite specification.

5.5.3.1.8 Test bitstream #GE-10

Specification: Bitstream with only intra macroblocks using only the DC coefficient and predicted macroblocks having no DCT coefficients. Reconstructed motion vectors used for predicting both luminance and chrominance have all possible combinations of half-sample and full-sample values, both for the horizontal and

the vertical coordinates, and all those combinations are used for each prediction mode in both progressive and interlaced coded VOPs.

Functional stage: MCP

Purpose: Check that decoder implements motion compensation stages with full accuracy in all cases. Except for reconstruction of Intra DC blocks, the test does not involve other decoder functions such as IDCT, inverse quantization and mismatch control. When a static decoder test is performed using the static test technique described in this document, the decoder under test shall reconstruct samples identical to those reconstructed by a reference decoder for all predicted macroblocks.

5.5.3.1.9 Test bitstream #GE-11

Specification: Flat distribution of VLC events (worst case for constant rate symbolic VLD's) on B- and P-VOPs. Number of MB/s and bitrate are the maximum allowed for the profile-and-level combination.

Functional stage: VLD

Purpose: Check that decoder does not rely on statistically low count of symbols over global areas to meet real-time constraints.

5.5.3.1.10 Test bitstream #GE-12

Specification: Bursty case for number of bits per macroblock with different burst location within VOP (top, bottom), followed by Bi-directional macroblocks. All motion vectors with half-sample components. Macroblocks outside the burst concentration have all bi-directional prediction. Number of MB/s and bitrate are the maximum allowed for the profile-and-level combination. Half-sample in both the horizontal and vertical directions, luminance and chrominance blocks. Maximize number of prediction blocks required to reconstruct a macroblock.

Functional stage: VLD and prediction bandwidth

Purpose: Check that decoder does not rely upon statistically small number of coded bits over local areas.

5.5.3.1.11 Test bitstream #GE-13

Specification: A series of consecutive progressively coded P-VOPs. As many half-sample components as possible in both the horizontal and vertical directions, luminance and chrominance blocks. Number of MB/s and bitrate are the maximum allowed for the profile-and-level combination. Maximize number of prediction blocks required to reconstruct a macroblock.

Functional stage: prediction bandwidth

Purpose: Check that decoder handles largest prediction bandwidth with progessively coded P-VOPs. This test is somehow similar to Test bitstream #3, except that it uses progressive VOPs.

5.5.3.1.12 Test bitstream #GE-14

Specification: A bitstream with a series of consecutive progressively coded B-VOPs with bi-directional macroblock motion compensation. Sequence contains many consecutive B-VOPs. Number of MB/s and bitrate are the maximum allowed for the profile-and-level combination. Use half-sample prediction in both the horizontal and vertical directions, for all luminance and chrominance blocks. Maximize number of prediction blocks required to reconstruct a macroblock.

Functional stage: prediction bandwidth

Purpose: Check that decoder can cope with this case of worst case bandwidth. This test is somehow similar to Test bitstream #1, except that it uses progressive VOPs.

5.5.3.1.13 Test bitstream #GE-16

Specification: Short header bitstream. Luminance sample rate and bitrate are the maximum allowed for ITU-T H.263 bitstream.

Functional stage: overall

Purpose: Check that decoder can decode short header (ITU-T H.263) bitstreams.

5.5.3.1.14 Test bitstream #GE-18

Specification: Low delay sequence with skipped VOPs. Number of MB/s and bitrate are the maximum allowed for the profile-and-level combination.

Functional stage: controller

Purpose: Check that decoder is capable of decoding low delay mode and knows how to recognize and deal with skipped VOPs and buffer underflows in the VBV model.

5.5.3.1.15 Test bitstream #GE-19

Specification: A bitstream implementing a test close to the IEEE 1180 IDCT mismatch test, to test the decoder's IDCT statistical accuracy. Can be done using I-VOPs with a flat custom quantization matrix with all 16, and a quantizer value of 1. Use whatever number of VOPs are required to satisfy statistic count. Note that because of saturation in [0, 255], the test cannot emulate exactly the IEEE 1180 IDCT test.

Functional stage: IDCT

Purpose: Check IDCT decoder accuracy. This is not a drift test since all macroblocks are of type Intra.

5.5.3.1.16 Test bitstream #GE-20

Specification: Bitstream causing maximum saturation of the inverse quantization by creating the greatest amplitude combinations of macroblock quantization (quantizer value 31), visual weighting matrix (value 2^n), and DCT coefficient (value -2^{n+3} or 2^{n+3}), where n is the maximum allowed number of bits per pixel for the profile-and-level combination. MPEG-2-style quantisation is used.

Functional stage: inverse quantization

Purpose: Test that decoder implements properly the saturation of the inverse quantization (before the mismatch control).

5.5.3.1.17 Test bitstream #GE-21

Specification: Bitstream causing maximum saturation of the inverse quantization by creating the greatest amplitude combinations of macroblock quantization (quantizer value 31), visual weighting matrix (value 2^n), and DCT coefficient (value -2^{n+3} or 2^{n+3}), where n is the maximum allowed number of bits per pixel for the profile-and-level combination. H.263-style quantisation is used.

Functional stage: inverse quantization

Purpose: Test that decoder implements properly the saturation of the inverse quantization (before the mismatch control).

5.5.3.1.18 Test bitstream #GE-22

Specification: Bitstream causing large positive sample domain coefficients f[y][x] (e.g., 255) added to large predicted values p[y][x] (e.g., 255), or large negative sample domain coefficients f[y][x] (e.g., -256) added to small predicted values p[y][x] (e.g., 0).

Functional stage: addition of the output of IDCT f[y][x] to the predicted values p[y][x] and saturation of the result to the range $[0, 2^n]$.

Purpose: Test that decoder implements properly the addition of the output of IDCT f[y][x] to the predicted values p[y][x] and saturation of the result to the range $[0, 2^n]$.

5.5.3.1.19 Test bitstream #GE-23

Specification: A bitstream with I-, P- and B-VOPs, with motion vectors that are as large as permitted by the profile-and-level combination.

Functional stage: reconstruction of motion vectors, MCP, control

Purpose: Check that decoder implements motion compensation properly when motion vectors are very large.

5.5.3.1.20 Test bitstream #GE-24

Specification: A bitstream with quantizer matrices (intra and non-intra, and if permitted, chroma matrices too). Matrices are not symmetrical (e.g., matrix coefficients are random numbers in the range $[1, 2^n]$). If permitted, use of both scanning orders.

Functional stage: quantizer matrix download, matrix scanning.

Purpose: Check that decoder can download properly quantizer matrices and that it uses of correct scanning of the matrices (i.e. not transposed).

5.5.3.1.21 Test bitstream #GE-25

Specification: A bitstream in which the output of the non-saturated integer number mathematical IDCT f ' (x, y), as defined in Annex A of ISO/IEC 14496-2, has large absolute values but values within the range $[-2^n-2^{n-1}, 2^n+2^{n-1}-1]$ for each coded block, where n is the maximum allowed number of bits per pixel for the profile-and-level combination.

Functional stage: IDCT

Purpose: Check that IDCT decoder accuracy meets the requirements defined in Annex A of ISO/IEC 14496-2. The peak error for a compliant decoder shall be less or equal to than 2 when decoding this bitstream. Note that for blocks where f ' (x, y) has values within the range [-300, 300], decoders that have a peak error larger than 1 may not be compliant with the IEEE 1180 IDCT specification.

5.5.3.1.22 Test Bitstream #MHH-1

Specification: This bitstream exercises all different horizontal half-pel motion vector values for vop_fcode_forward =1. The vertical motion displacement toggles between 0 and 1 half-pels.

Functional stage: Motion vector decoding; motion compensation.

Purpose: To check that for vop_fcode_forward=1 the decoder properly handles the full range of Px, MVDx, and MVx for 16x16 block size half-pel motion compensated rectangular P-VOPs.

5.5.3.1.23 Test Bitstream #MHH-2

Specification: This bitstream exercises all different horizontal half-pel motion vector values for vop_fcode_forward =2. The vertical motion displacement toggles between 0 and 1 half-pels.

Functional stage: Motion vector decoding; motion compensation.

Purpose: To check that for vop_fcode_forward=2 the decoder properly handles the full range of Px, MVDx, and MVx for 16x16 block size half-pel motion compensated rectangular P-VOPs.

5.5.3.1.24 Test Bitstream #MHH-3

Specification: This bitstream exercises all different horizontal half-pel motion vector values for vop_fcode_forward =3. The vertical motion displacement toggles between 0 and 1 half-pels.

Functional stage: Motion vector decoding; motion compensation.

Purpose: To check that for vop_fcode_forward=3 the decoder properly handles the full range of Px, MVDx, and MVx for 16x16 block size half-pel motion compensated rectangular P-VOPs.

5.5.3.1.25 Test Bitstream #MHH-4

Specification: This bitstream exercises all different horizontal half-pel motion vector values for vop_fcode_forward =4. The vertical motion displacement toggles between 0 and 1 half-pels.

Functional stage: Motion vector decoding; motion compensation.

Purpose: To check that for vop_fcode_forward=4 the decoder properly handles the full range of Px, MVDx, and MVx for 16x16 block size half-pel motion compensated rectangular P-VOPs.

5.5.3.1.26 Test Bitstream #MHH-5

Specification: This bitstream exercises all different horizontal half-pel motion vector values for vop_fcode_forward =5. The vertical motion displacement toggles between 0 and 1 half-pels.

Functional stage: Motion vector decoding; motion compensation.

Purpose: To check that for vop_fcode_forward=5 the decoder properly handles the full range of Px, MVDx, and MVx for 16x16 block size half-pel motion compensated rectangular P-VOPs.

5.5.3.1.27 Test Bitstream #MHH-6

Specification: This bitstream exercises all different horizontal half-pel motion vector values for vop_fcode_forward =6. The vertical motion displacement toggles between 0 and 1 half-pels.

Functional stage: Motion vector decoding; motion compensation.

Purpose: To check that for vop_fcode_forward=6 the decoder properly handles the full range of Px, MVDx, and MVx for 16x16 block size half-pel motion compensated rectangular P-VOPs.

5.5.3.1.28 Test Bitstream #MHH-7

Specification: This bitstream exercises all different horizontal half-pel motion vector values for vop_fcode_forward =7. The vertical motion displacement toggles between 0 and 1 half-pels.

Functional stage: Motion vector decoding; motion compensation.

Purpose: To check that for vop_fcode_forward=7 the decoder properly handles the full range of Px, MVDx, and MVx for 16x16 block size half-pel motion compensated rectangular P-VOPs.

5.5.3.1.29 Test Bitstream #MVH-1

Specification: This bitstream exercises all different vertical half-pel motion vector values for vop_fcode_forward =1. The horizontal motion displacement toggles between 0 and 1 half-pels.

Functional stage: Motion vector decoding; motion compensation.

Purpose: To check that for vop_fcode_forward=1 the decoder properly handles the full range of Py, MVDy, and MVy for 16x16 block size half-pel motion compensated rectangular P-VOPs.

5.5.3.1.30 Test Bitstream #MVH-2

Specification: This bitstream exercises all different vertical half-pel motion vector values for vop_fcode_forward =2. The horizontal motion displacement toggles between 0 and 1 half-pels.

Functional stage: Motion vector decoding; motion compensation.

Purpose: To check that for vop_fcode_forward=2 the decoder properly handles the full range of Py, MVDy, and MVy for 16x16 block size half-pel motion compensated rectangular P-VOPs.

5.5.3.1.31 Test Bitstream #MVH-3

Specification: This bitstream exercises all different vertical half-pel motion vector values for vop_fcode_forward =3. The horizontal motion displacement toggles between 0 and 1 half-pels.

Functional stage: Motion vector decoding; motion compensation.

Purpose: To check that for vop_fcode_forward=3 the decoder properly handles the full range of Py, MVDy, and MVy for 16x16 block size half-pel motion compensated rectangular P-VOPs.

5.5.3.1.32 Test Bitstream #MVH-4

Specification: This bitstream exercises all different vertical half-pel motion vector values for vop_fcode_forward =4. The horizontal motion displacement toggles between 0 and 1 half-pels.

Functional stage: Motion vector decoding; motion compensation.

Purpose: To check that for vop_fcode_forward=4 the decoder properly handles the full range of Py, MVDy, and MVy for 16x16 block size half-pel motion compensated rectangular P-VOPs.

5.5.3.1.33 Test Bitstream #MVH-5

Specification: This bitstream exercises all different vertical half-pel motion vector values for vop_fcode_forward =5. The horizontal motion displacement toggles between 0 and 1 half-pels.

Functional stage: Motion vector decoding; motion compensation.

Purpose: To check that for vop_fcode_forward=5 the decoder properly handles the full range of Py, MVDy, and MVy for 16x16 block size half-pel motion compensated rectangular P-VOPs.

5.5.3.1.34 Test Bitstream #MVH-6

Specification: This bitstream exercises all different vertical half-pel motion vector values for vop_fcode_forward =6. The horizontal motion displacement toggles between 0 and 1 half-pels.

Functional stage: Motion vector decoding; motion compensation.

Purpose: To check that for vop_fcode_forward=6 the decoder properly handles the full range of Py, MVDy, and MVy for 16x16 block size half-pel motion compensated rectangular P-VOPs.

5.5.3.1.35 Test Bitstream #MVH-7

Specification: This bitstream exercises all different vertical half-pel motion vector values for vop_fcode_forward =7. The horizontal motion displacement toggles between 0 and 1 half-pels.

Functional stage: Motion vector decoding; motion compensation.

Purpose: To check that for vop_fcode_forward=7 the decoder properly handles the full range of Py, MVDy, and MVy for 16x16 block size half-pel motion compensated rectangular P-VOPs.

5.5.3.2 Test Bitstreams - Shape coding

Three classes of bit streams are defined to test shape coding. The first class applies to every profile@level for which shape coding is defined, and tests the correct interpretation of the syntax and semantics by the decoder. For the two other classes, a separate bitstream is required for each level, such as to test the conditions defined for each profile/level. Bitstreams of the second class contain shape information only, and bitstreams of the third class contain both shape and texture information. These bitstreams are generated using an encoder that makes a random decision whenever it has to make one.

NOTE — The precision of the arithmetic coder is defined by ISO/IEC 14496-2. Failure to comply with the defined precision will generally produce errors and de-synchronize the decoder. Also the decoded binary shape must always exactly match with the output produced by the reference software. Failure to do so will most likely result in de-synchronization of the decoder

Table 7 – Description of Bitstream Class for Shape Coding Conformance

Class	Profile	Description
1	1 for all profile@level's	shape only bitstream generated "by hand" tests 1024 contexts of intra-CAE, 512 contexts of inter-CAE, 256 contexts for up-sampling, shape motion vectors
2	1 for each profile@level	Shape only bitstream generated by random decision maker general test of shape coding
3	1 for each profile@level	Shape and texture bitstream generated by random decision maker general test of shape/texture coding in particular, tests padding and prediction of shape motion vectors from texture motion vectors for I, P and B frames

The input used to generate bitstreams of classes 2 and 3 will consist of the concatenation of several typical test sequences.

Class 1:

5.5.3.2.1 Test Bitstream #SH-1

Specification: A series of consecutive I- and P-VOP with half of the macroblocks lying on the boundary, i.e. coded with the intra- and inter-CAE procedures. The bitstream is designed such as to use every entry in all look-up tables defined by binary shape coding. Test conditions are the maximum allowed for the profile@level combination. This bitstream contains binary shape only information.

Functional stage: intra-CAE, inter-CAE, up-sampling and down-sampling, MB bandwidth

Purpose: Check 1024 contexts of intra-CAE, 512 contexts of inter-CAE, 256 contexts for up-sampling, and down-sampling.

Class 2:

5.5.3.2.2 Test Bitstream #SH-2

Specification: A series of consecutive I- and P-VOPs with binary shape only coding. The bitstream production is controlled by a random decision maker. This bitstream is made under the condition of core profile @ level 1.

Functional stage: MV for shape, BAB type coding, MB bandwith, reference memory bandwidth

Purpose: Check the general case of testing binary shape coding with proper test sequence for for a given profile @ level structure.

5.5.3.2.3 Test Bitstream #SH-3

Specification: A series of consecutive I- and P-VOPs with binary shape only coding. The bitstream production is controlled by a random decision maker. This bitstream is made under the condition of core profile @ level 2.

Functional stage: MV for shape, BAB type coding, MB bandwith, reference memory bandwidth

Purpose: Check the general case of testing binary shape coding with proper test sequence for for a given profile @ level structure.

5.5.3.2.4 Test Bitstream #SH-4

Specification: A series of consecutive I- and P-VOPs with binary shape only coding. The bitstream production is controlled by a random decision maker. This bitstream is made under the condition of main profile @ level 2.

Functional stage: MV for shape, BAB type coding, MB bandwith, reference memory bandwidth

Purpose: Check the general case of testing binary shape coding with proper test sequence for a given profile @ level structure.

5.5.3.2.5 Test Bitstream #SH-5

Specification: A series of consecutive I- and P-VOPs with binary shape only coding. The bitstream production is controlled by a random decision maker. This bitstream is made under the condition of main profile @ level 3.

Functional stage: MV for shape, BAB type coding, MB bandwith, reference memory bandwidth

Purpose: Check the general case of testing binary shape coding with proper test sequence for a given profile @ level structure.

5.5.3.2.6 Test Bitstream #SH-6

Specification: A series of consecutive I- and P-VOPs with binary shape only coding. The bitstream production is controlled by a random decision maker. This bitstream is made under the condition of main profile @ level 4.

Functional stage: MV for shape, BAB type coding, MB bandwith, reference memory bandwidth

Purpose: Check the general case of testing binary shape coding with proper test sequence for a given profile @ level structure

Class 3:

5.5.3.2.7 Test Bitstream #SH-7

Specification: A series of consecutive I- and P-VOPs with binary shape and texture. The bitstream generation is controlled by a random decision maker. This bitstream is made under the condition of core profile @ level 1.

Functional stage: prediction of shape MV from texture MV

Purpose: check the general case of shape and texture coding. In particularly, tests padding and prediction of shape motion vectors from texture motion vectors with proper test sequence for a given profile @ level structure.

5.5.3.2.8 Test Bitstream #SH-8

Specification: A series of consecutive I- and P-VOPs with binary shape and texture. The bitstream generation is controlled by a random decision maker. This bitstream is made under the condition of core profile @ level 2.

Functional stage: prediction of shape MV from texture MV

Purpose: check the general case of shape and texture coding. In particularly, tests padding and prediction of shape motion vectors from texture motion vectors with proper test sequence for a given profile @ level structure.

5.5.3.2.9 Test Bitstream #SH-9

Specification: A series of consecutive I- and P-VOPs with binary shape and texture. The bitstream generation is controlled by a random decision maker. This bitstream is made under the condition of main profile @ level 2.

Functional stage: prediction of shape MV from texture MV

Purpose: check the general case of shape and texture coding. In particularly, tests padding and prediction of shape motion vectors from texture motion vectors with proper test sequence for a given profile @ level structure.

5.5.3.2.10 Test Bitstream #SH-10

Specification: A series of consecutive I- and P-VOPs with binary shape and texture. The bitstream generation is controlled by a random decision maker. This bitstream is made under the condition of main profile @ level 3.

Functional stage: prediction of shape MV from texture MV

Purpose: check the general case of shape and texture coding. In particularly, tests padding and prediction of shape motion vectors from texture motion vectors with proper test sequence for a given profile @ level structure.

5.5.3.2.11 Test Bitstream #SH-11

Specification: A series of consecutive I- and P-VOPs with binary shape and texture. The bitstream generation is controlled by a random decision maker. This bitstream is made under the condition of main profile @ level 4.

Functional stage: prediction of shape MV from texture MV

Purpose: check the general case of shape and texture coding. In particularly, tests padding and prediction of shape motion vectors from texture motion vectors with proper test sequence for a given profile @ level structure.

5.5.3.3 Test Bitstreams – Scalability

5.5.3.3.1 Test Bistream SCS-1

Specification: The enhancement layer bitstream contains VOP coded with ref_select_code = `00` in B-VOP and ref_select_code == `11` in P-VOP. The base layer bitstream contains P-VOP with skip macroblock.

The upsampling factors are set as follows.

horizontal_sampling_factor_n 16

horizontal_sampling_factor_m 1

vertical_sampling_factor_n 16

vertical_sampling_factor_m 1

Functional stage: Prediction process from base layer

Purpose: This bitstream tests prediction process from base layer, i.e. Temporally coincident VOP in the reference layer (no motion vectors).

5.5.3.3.2 Test Bistream SCS-2

Specification: The enhancement layer bitstream contains VOP coded with ref_select_code = `11` in B-VOP and ref_select_code == `11` in P-VOP. The base layer bitstream contains P-VOP with skip macroblock.

The upsampling factors are set as follows.

horizontal_sampling_factor_n 16

horizontal_sampling_factor_m 1

vertical_sampling_factor_n 16

vertical_sampling_factor_m 1

Functional stage: Prediction process from the enhancement layer

Purpose: This bitstream tests prediction process from enhancement layer. i.e. Most recently decoded enhancement VOP of the same layer . This bitstream also tests macroblock skipping rule in enhancement layer.

5.5.3.3.3 Test bitstream SCS-3

Specification: The enhancement layer bitstream contains VOP coded with ref_select_code = `00` in B-VOP and ref_select_code = `11` in P-VOP. The base layer bitstream contains P-VOP with skip macroblock.

The upsampling factors are set as follows.

horizontal_sampling_factor_n 16

horizontal_sampling_factor_m 1

vertical_sampling_factor_n 16

vertical_sampling_factor_m 1

Functional stage: Interpolate prediction process

Purpose: This bitstream tests interpolate prediction process from enhancement layer and base layer.

5.5.3.3.4 Test bitstream SCS-4

Specification: The bitstream has I and P-VOP in base layer and only B-VOP in enhancement layer. The base layer is compliant bitstream of Simple profile and at least one skipped MB is included in a P-VOP. The ref_select_code =“11“ of B-VOP is used for enhancement layer.

Function stage: Prediction process from the enhancement layer

Purpose: The purpose of this bitstream is to verify temporal scalability in the case of ref_select_code=“11“ of B-VOP. This bitstream also tests macroblock skipping rule in enhancement layer.

5.5.3.3.5 Test bitstream SCS-5

Specification: The bitstream has I and P-VOP in base layer and P and B-VOP in enhancement layer. The base layer is compliant bitstream of Simple profile and at least one skipped MB is included in a P-VOP. The ref_select_code = “01“ in B-VOP and ref_select_code = “01“ in P-VOP are used for enhancement layer.

Function stage: Prediction process from the enhancement layer

Purpose: The purpose of this bitstream is to verify temporal scalability in the case of ref_select_code=“01“ in B-VOP and ref_select_code = “01“ in P-VOP. This bitstream also tests macroblock skipping rule in enhancement layer.

5.5.3.3.6 Test bitstream SCS-6

Specification: The bitstream has I and two P-VOPs in base layer and P and B-VOP in enhancement layer. The base layer is compliant bitstream of Simple profile and at least one skipped MB is included in a P-VOP. The ref_select_code = “10“ in B-VOP and ref_select_code =“10“ in P-VOP are used for enhancement layer.

Function stage: Prediction process from the enhancement layer

Purpose: The purpose of this bitstream is to verify temporal scalability in the case of ref_select_code=“10“ in B-VOP and ref_select_code =“10“ in P-VOP. This bitstream also tests macroblock skipping rule in enhancement layer.

5.5.3.3.7 Test bitstream SCS-7

Specification: The bitstream has only one I-VOP in base layer and two P-VOPs in enhancement layer. The base layer is compliant bitstream of Simple profile and at least one skipped MB is included in a P-VOP. The ref_select_code =“00“ and “01“ in P-VOP is used for enhancement layer.

Function stage: Prediction process from the enhancement layer

Purpose: The purpose of this bitstream is to verify temporal scalability in the case of ref_select_code = “01“ and “00“ in P-VOP. This bitstream also tests macroblock skipping rule in enhancement layer.

Performance Tests

5.5.3.3.8 Test Bistream SCS-8

Specification: The bitstream contains VOP coded with ref_select_code = `11` in enhancement layer P-VOPs and ref_select_code = `00` in enhancement layer B-VOPs.

This bitstream has bitrate and Macroblocks per second with the upper bound value of L1 in Simple scalable profile.

Functional stage: Performance of enhancement layer decoder

Purpose: This bitstream tests performance of enhancement layer decoder. This bitstream put stress for enhancement layer decoder in L1.

5.5.3.3.9 Test Bistream SCS-9

Specification: The bitstream contains VOP coded with ref_select_code = `11` in enhancement layer P-VOPs and ref_select_code = `00` in enhancement layer B-VOPs.

This bitstream has bitrate and Macroblocks per second with the upper bound value of L2 Simple scalable profile.

Functional stage: Performance of enhancement layer decoder

Purpose: This bitstream tests performance of enhancement layer decoder. This bitstream put stress for enhancement layer decoder in L2.

5.5.3.3.10 Test bitstream SCS-10

Specification: The base layer is compliant bitstream of Simple profile. The ref_select_code = “01“ in B-VOP and ref_select_code = “01“ in P-VOP are used for enhancement layer. The max number of bitrate and Macroblock per second satisfy those of SSP@L1

Function stage: Performance of enhancement layer decoder

Purpose: The purpose of this bitstream is to verify a performance of enhancement layer decoder. The bitstream put stress for enhancement layer decoder in SSP@L1.

5.5.3.3.11 Test bitstream SCS-11

Specification: The base layer is compliant bitstream of Simple profile. The ref_select_code = “01“ in B-VOP and ref_select_code = “01“ in P-VOP are used for enhancement layer. The max number of bitrate and Macroblock per second satisfy those of SSP@L2

Function stage: Performance of enhancement layer decoder

Purpose: The purpose of this bitstream is to verify a performance of enhancement layer decoder. The bitstream put stress for enhancement layer decoder in SSP@L2.

5.5.3.4 Conformance test conditions for scalability in the Core Object

For the Core Object, the following functional and perfomance tests have to be applied for testing of decoder conformance.

Functional Tests

5.5.3.4.1 Test bitstream SCC-1

Specification: The bitstream has I and P-VOP in base layer and only one P-VOP in enhancement layer. The base layer is compliant bitstream of Core profile without B-VOP and the reconstructed image should be rectangular VOP. The ref_select_code = “10“ in arbitrary shaped P-VOP with enhancement_type = “1“ is used for enhancement layer.

Function stage: Prediction process from the enhancement layer

Purpose: The purpose of this bitstream is to verify temporal scalability with a rectangular VOP in base layer and arbitrary shape in enhancement layer. In addtion to that, ref_select_code=“10“ in P-VOP case is verified.

5.5.3.4.2 Test bitstream SCC-2

Specification: The bitstream has I and P-VOP in base layer and only one P-VOP in enhancement layer. The base layer is compliant bitstream of Core profile without B-VOP and the reconstructed image should be

arbitrary shaped VOP. The ref_select_code = “10“ in arbitrary shaped P-VOP with enhancement_type = “0“ is used for enhancement layer.

Function stage: Prediction process from the enhancement layer

Purpose: The purpose of this bitstream is to verify temporal scalability with arbitrary shaped VOP in both base and enhancement layer with enhancement_type = “0“. In addtion to that, ref_select_code=“10“ in P-VOP case is verified.

5.5.3.4.3 Test bitstream SCC-3

Specification: The bitstream has I and P-VOP in base layer and only one P-VOP in enhancement layer. The base layer is compliant bitstream of Core profile without B-VOP and the reconstructed image should be arbitrary shaped VOP. The ref_select_code = “10“ in arbitrary shaped P-VOP with enhancement_type = “1“ is used for enhancement layer.

Function stage: Prediction process from the enhancement layer

Purpose: The purpose of this bitstream is to verify temporal scalability with arbitrary shaped VOP in both base and enhancement layer with enhancement_type = “1“. In addtion to that, ref_select_code=“10“ in P-VOP case is verified.

5.5.3.4.4 Test bitstream SCC-4

Specification: The bitstream has only I-VOP in base layer and two P-VOPs in enhancement layer. The base layer is compliant bitstream of Core profile without B-VOP and the reconstructed image should be rectangular VOP. The ref_select_code = “01“ and “00“ in arbitrary shaped P-VOP with enhancement_type = “1“ is used for enhancement layer.

Function stage: Prediction process from the enhancement layer

Purpose: The purpose of this bitstream is to verify temporal scalability with a rectangular VOP in base layer and arbitrary shape in enhancement layer. In addtion to that, ref_select_code=“01“ and “00“ in P-VOP case is verified.

5.5.3.4.5 Test bitstream SCC-5

Specification: The bitstream has only one I-VOP in base layer and two P-VOPs in enhancement layer. The base layer is compliant bitstream of Core profile without B-VOP and the reconstructed image should be arbitrary shaped VOP. The ref_select_code = “01“ and “00“ in arbitrary shaped P-VOP with enhancement_type = “0“ is used for enhancement layer.

Function stage: Prediction process from the enhancement layer

Purpose: The purpose of this bitstream is to verify temporal scalability with arbitrary shaped VOP in both base and enhancement layer with enhancement_type = “0“. In addtion to that, ref_select_code=“01“ and “00“ in P-VOP case is verified.

5.5.3.4.6 Test bitstream SCC-6

Specification: The bitstream has only one I-VOP in base layer and two P-VOPs in enhancement layer. The base layer is compliant bitstream of Core profile without B-VOP and the reconstructed image should be arbitrary shaped VOP. The ref_select_code = “01“ and “00“ in arbitrary shaped P-VOP with enhancement_type = “1“ is used for enhancement layer.

Function stage: Prediction process from the enhancement layer

Purpose: The purpose of this bitstream is to verify temporal scalability with arbitrary shaped VOP in both base and enhancement layer with enhancement_type = “1“. In addtion to that, ref_select_code=“01“ and “00“ in P-VOP case is verified.

Performance Tests

Common Conditions

NOTE — base must not include B-VOP

Both load_forward_shape and load_backward_shape should be zero.

5.5.3.4.7 Test bitstream SCC-7

Specification: The base layer is compliant bitstream of Core profile without B-VOP. The ref_select_code = “00“ and “01“ in P-VOP are used for enhancement layer. The max number of bitrate and Macroblock per second satisfy those of CP@L1.

Function stage: Performance of enhancement layer decoder

Purpose: This bitstream tests performance of enhancement layer decoder. This bitstream put stress for enhancement layer decoder in CP@L1.

5.5.3.4.8 Test bitstream SCC-8

Specification: The base layer is compliant bitstream of Core profile without B-VOP. The ref_select_code = “00“ and “01“ in P-VOP are used for enhancement layer. The max number of bitrate and Macroblock per second satisfy those of CP@L2.

Function stage: Performance of enhancement layer decoder

Purpose: This bitstream tests performance of enhancement layer decoder. This bitstream put stress for enhancement layer decoder in CP@L2.

5.5.3.5 Test Bstreams - Error resilience

5.5.3.5.1 Test bitstream #er-1

Specification: The use of resynchronisation markers in a video bistream.

Functional stage: bitstream parser

Purpose: To ensure that the decoder can successfully decode video with both large and small spacings between resynchronisation markers and can parse the HEC field of the video packet header.

5.5.3.5.2 Test bitstream #er-2

Specification: The use of data partitioning mode in a video bistream.

Functional stage: bitstream parser

Purpose: To ensure that the decoder can successfully decode video with both large and small spacings between resynchronisation markers when data partitioning is used. The decoder should be stressed by using the maximum allowed spacings between resynchronisation markers.

5.5.3.5.3 Test bitstream #er-3

Specification: The use of data partitioning and reversible variable length codes in a video bistream.

Functional stage: bitstream parser

Purpose: To ensure that the decoder can successfully decode video with both large and small spacings between resynchronisation markers when data partitioning and reversible variable length codes are used. The decoder should be stressed by using the maximum allowed spacings between resynchronisation markers.

5.5.3.6 Test Bitstreams - Scalable still texture

5.5.3.6.1 Test bitstream #ss-XX

Specification The bitstream are generated using single_quant mode with quantization step size 1, without scalability start codes and maximum levels. The following cases are tested:

Table 8 – Tested Cases for Scalable Still Texture

Case	Integer/float	Default /Downloadable	Level to be tested
1	I	Default	0, 1, 2
2	F	Default	2
3	I	Down	1,2
4	F	Down	2

Functional stage: IDWT

Purpose: To test a decoder for conformance with the regard of scalable still texture profile, the above bitstream are used to verify the accuracy of the IDWT

Specification: The bitstream are generated using default integer wavelet and maximum number of levels.

Table 9 – Parameters of Test Cases

Case	quantization	Scanning	start_code	Spatial Scalability	SNR scalability
1	SQ	TD	off	1	1
2	SQ	BB	off	Max	1
3	SQ	BB	on	Max	1
4	MQ	TD	off	1	Max
5	MQ	TD	on	1	Max
6	MQ	BB	off	Max	Max
7	MQ	BB	off	Max	Max
8	BQ	TD	on	Max	Max
9	BQ	BB	on	Max	Max

Functional stage: Scalable texture coding

Purpose: To test a decoder for conformance with the regard of scalability of still texture profile, the above bitstream are decoded at first layer of spatial/SNR scalability, last SNR layer of first spatial scalability, first, middle and last SNR layers of the middle spatial scalability layer and finally at first and last SNR layer of final spatial scalability.

5.5.3.7 Test Bitstreams - Sprites

5.5.3.7.1 Test bitstream #sp1

Specification: basic sprite, stationary warping (no_of_sprite_warping_points == 0).

Functional stage: warping.

Purpose: Test whether the warping function ($F(i, j)$, $G(i, j)$, $F_c(i_c, j_c)$, and $G_c(i_c, j_c)$ specified in subclause 7.8.5 of ISO/IEC 14496-2) is implemented conforming to the accuracy restrictions (no errors).

5.5.3.7.2 Test bitstream #sp2

Specification: basic sprite, translational warping (no_of_sprite_warping_points == 1), half pixel accuracy (sprite_warping_accuracy == “1/2 pixel”).

Functional stage: warping, pixel value interpolation, and real time decoding.

Purpose: Test whether the warping function ($F(i, j)$, $G(i, j)$, $F_c(i_c, j_c)$, and $G_c(i_c, j_c)$ specified in subclause 7.8.5 of ISO/IEC 14496-2) is implemented conforming to the accuracy restrictions (no errors).

5.5.3.7.3 Test bitstream #sp3

Specification: basic sprite, isotropic warping (no_of_sprite_warping_points == 2), quarter pixel accuracy (sprite_warping_accuracy == “1/4 pixel”).

Functional stage: warping, pixel value interpolation

Purpose: Test whether the warping function ($F(i, j)$, $G(i, j)$, $F_c(i_c, j_c)$, and $G_c(i_c, j_c)$ specified in sublause 7.8.5 of ISO/IEC 14496-2) is implemented conforming to the accuracy restrictions (no errors).

5.5.3.7.4 Test bitstream #sp4

Specification: basic sprite, affine warping (no_of_sprite_warping_points == 3), 1/8 pixel accuracy (sprite_warping_accuracy == “1/8 pixel”).

Functional stage: Warping, pixel value interpolation

Purpose: Test whether the warping function ($F(i, j)$, $G(i, j)$, $F_c(i_c, j_c)$, and $G_c(i_c, j_c)$ is implemented conforming to the accuracy restrictions (no errors).

5.5.3.7.5 Test bitstream #sp5

Specification: basic sprite, perspective warping (no_of_sprite_warping_points == 4), 1/16 pixel accuracy (sprite_warping_accuracy == “1/16 pixel”).

Functional stage: Warping, pixel value interpolation

Purpose: Test whether the warping function ($F(i, j)$, $G(i, j)$, $F_c(i_c, j_c)$, and $G_c(i_c, j_c)$ is implemented conforming to the accuracy restrictions (+1/-1 errors).

5.5.3.7.6 Test bitstream #sp6

Specification: low-latency sprite, affine warping (no_of_sprite_warping_points == 3), half pel accuracy (sprite_warping_accuracy == “1/2 pixel”)

Functional stage: real time decoding, IDCT

Purpose: Test whether real time decoding of low-latency sprite bitstreams conforming to the VBV and VCV buffer model is possible. Test whether the sprite is decoded conforming to the IDCT accuracy restrictions (+1/-1 errors).

5.5.4 Implementation of the static test

For each bitstream of the test suite, the following operations are performed.

The bitstream is decoded by the decoder under test. All the samples reconstructed by the decoder under test are captured and stored for future use.

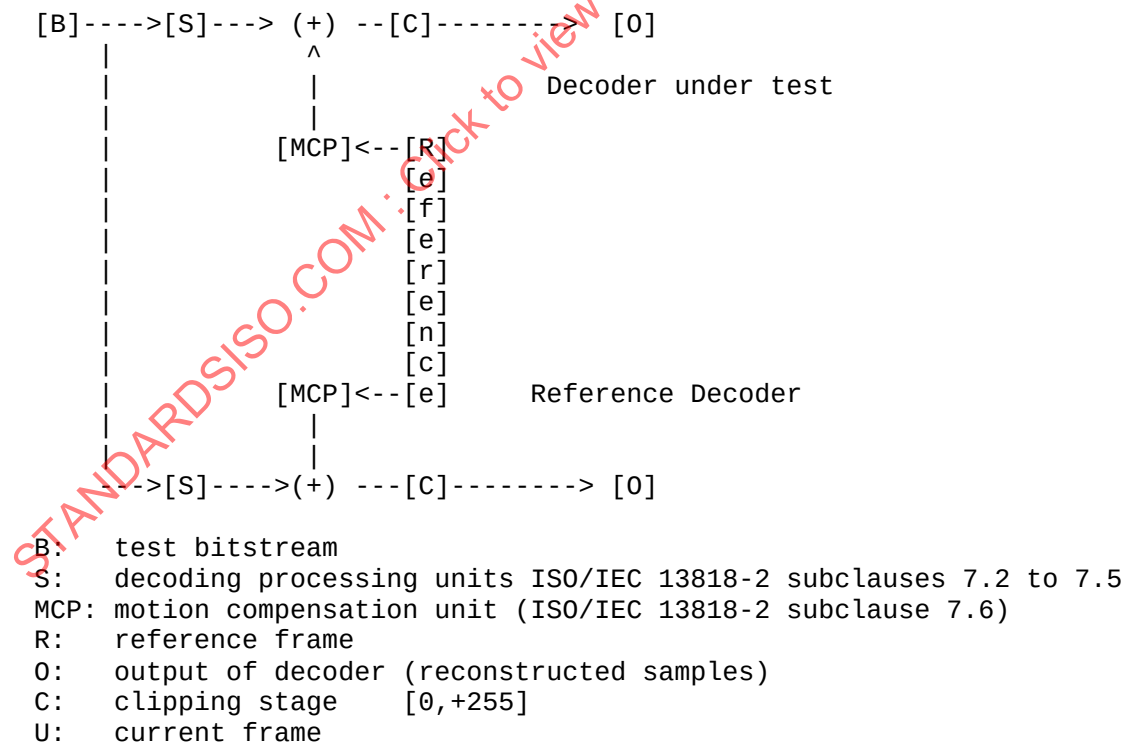
The bitstream is then decoded by the reference decoder as follows:

Before decoding each P- or B-picture or enhancement layers of multi-layer objects, the frame buffers of the reference decoder are initialized with the reconstructed samples captured from the decoder under test that correspond to those reference frames.

This method called “frame buffer intercept method” guarantees that the decoder under test and the reference decoder use the same reference frames, and therefore that mismatch does not accumulate. See Figure 3.

Then the samples reconstructed by the reference decoder are captured for each reconstructed picture, and compared to those reconstructed by the decoder under test (previously captured) for the same picture.

This methodology guarantees that there cannot be accumulations of errors, and that the difference observed for each sample only involves one IDCT process.



NOTE - R is kept identical in both the Reference and Test Decoders.

Figure 3 — Frame buffer intercept method

5.5.5 Implementation of the dynamic test

The dynamic test is often easier to perform on the complete decoder system, which includes a systems decoder, a video decoder and a display process. It is possible to record the output of the display process and to check that display order and timing of fields or frames are correct. However, since the display process is not within the normative scope of ISO/IEC 14496-2, there may be cases where the output of the display process is wrong even though the video decoder is compliant. In this case, the output of the video decoder itself (before the display process) must be captured in order to perform the dynamic tests on the video decoder.

In particular the field or frame order and timing shall be correct, field parity must be accurate (e.g. the first output field of interlaced frame with top_field_first equals to zero must be the bottom field), and that fields or frames that are coded as being repeated are indeed repeated at the output of the decoding process.

5.5.6 Decoder conformance

In order for a decoder of a particular profile-and-level to claim compliance to the standard described by this document, the decoder shall pass successfully both the static test defined in 5.1 and the dynamic test defined in 5.2 with all the bitstreams of the normative test suite specified for testing decoders of this particular profile-and-level.

Tables in subsequent subclauses define the normative test suites for each profile-and-level combination. The test suite for a particular profile-and-level combination is the list of bitstreams that are marked with a 'D', 'S' or 'X' in the column corresponding to that profile-and-level combination.

'D' indicates that the bitstream is designed to test the dynamic conformance of the decoder.

'S' indicates that the bitstream is designed to test the static conformance of the decoder.

'X' indicates that the bitstream is designed to test both the dynamic and static conformance of the decoder.

Bitstream specification indicates the test bitstream specification used for each bitstream.

When the test suite for a profile-and-level combination does not include any bitstream of this same profile-and-level, it is not possible to test adequately compliance to the standard for decoders of that profile-and-level.

5.5.7 Normative Test Suites for Simple, Simple Scalable, Core, Main and N-Bit profile

Legend:

S – Bitstream is intended for functional test

D – Bitstream is intended for dynamic test

X – Bitstream is for functional and dynamic test

Table 10 — Normative Test Suites for Simple, Simple Scalable, Core, Main and N-Bit profile

Categories	Bitstream	Donated by	Bitstreams Name	Simple			Simple Sclable (Enhance)		Core		Main			N-Bit	Scalable Texture		
				L1	L2	L3	L1	L2	L1	L2	L2	L3	L4	L2	L1	L2	L3
General	GE-1	GI	vcon-ge1.cmp									S					
	GE-2	GI	vcon-ge2.cmp									S					
	GE-3	GI	vcon-ge3.cmp									S					
	GE-4	GI	vcon-ge4.cmp									S					
	GE-6	GI	vcon-ge6.cmp									S					
	GE-8	GI	vcon-ge8.cmp			S											
	GE-10	GI	vcon-ge10.cmp									S					
	GE-11	GI	vcon-ge11.cmp									S					
	GE-12	GI	vcon-ge12.cmp									S					
	GE-13	Mitsubishi	vcon-ge13-L1.bits	S													
	GE-13	Mitsubishi	vcon-ge13-L2.bits		S												

Categories	Bitstream	Donated by	Bitstreams Name	Simple			Simple Sclable (Enhance)		Core		Main			N-Bit	Scalable Texture		
				L1	L2	L3	L1	L2	L1	L2	L2	L3	L4	L2	L1	L2	L3
	GE-13	Mitsubishi	vcon-ge13-L3.bits			S											
	GE-14	GI	vcon-ge14.cmp									S					
	GE-16	Mitsubishi	vcon-ge16-L1.bits	S													
	GE-16	Mitsubishi	vcon-ge16-L2.bits		S												
	GE-16	Mitsubishi	vcon-ge16-L3.bits			S											
	GE-18	GI	vcon-ge18.cmp			S											
	GE-19	GI	vcon-ge19.cmp									S					
	GE-20	GI	vcon-ge20.cmp									S					
	GE-21	GI	vcon-ge21.cmp									S					
	GE-22	GI	vcon-ge22.cmp									S					
	GE-23	GI	vcon-ge23.cmp									S					
	GE-24	GI	vcon-ge24.cmp			S											
	GE-25	GI	vcon-ge25.cmp			S											
	MHH-1	Sorenson	hlfpel1h.bits	X													
	MHH-2	Sorenson	hlfpel2h.bits	X													
	MHH-3	Sorenson	hlfpel3h.bits	X													
	MHH-4	Sorenson	hlfpel4h.bits	X													
	MHH-5	Sorenson	hlfpel5h.bits	X													
	MHH-6	Sorenson	hlfpel6h.bits	X													
	MHH-7	Sorenson	hlfpel7h.bits	X													
	MVH-1	Sorenson	hlfpel1v.bits	X													
	MVH-2	Sorenson	hlfpel2v.bits	X													
	MVH-3	Sorenson	hlfpel3v.bits	X													
	MVH-4	Sorenson	hlfpel4v.bits	X													
	MVH-5	Sorenson	hlfpel5v.bits	X													
	MVH-6	Sorenson	hlfpel6v.bits	X													
	MVH-7	Sorenson	hlfpel7v.bits	X													
Binary Shape	SH-1	Sony	Vcon-sh1.bits						S	S	S	S	S				
	SH-2	Sony	Vcon-sh2.bits						S								
	SH-3	Sony	Vcon-sh3.bits							S							
	SH-4	Sony	Vcon-sh4.bits								S						
	SH-5	Sony	Vcon-sh5.bits									S					
	SH-6	Sony	Vcon-sh6.bits										S				
	SH-7-1	Toshiba	vcon-sh7-1.cmp						S								
	SH-7-2	Toshiba	vcon-sh7-2.cmp						S								
	SH-8-1	Toshiba	vcon-sh8-1.cmp							S							
	SH-8-2	Toshiba	vcon-sh8-2.cmp							S							
	SH-9-1	Samsung	vcon-sh9-1.cmp								S						
	SH-9-2	Samsung	vcon-sh9-2.cmp								S						
	SH-10-1	Samsung	vcon-sh10-1.cmp									S					
	SH-10-2	Samsung	Vcon-sh10-2.cmp									S					
Scalability	SCS-1	Sony	vcon-scs1.bits	S	S	S											

Categories	Bitstream	Donated by	Bitstreams Name	Simple			Simple Sclable (Enhance)		Core		Main			N-Bit	Scalable Texture		
				L1	L2	L3	L1	L2	L1	L2	L2	L3	L4	L2	L1	L2	L3
	SCS-1_e	Sony	vcon-scs1_e.bits				S	S									
	SCS-2	Sony	vcon-scs2.bits	S	S	S											
	SCS-2_e	Sony	vcon-scs2_e.bits				S	S									
	SCS-3	Sony	vcon-scs3.bits	S	S	S											
	SCS-3_e	Sony	vcon-scs3_e.bits				S	S									
	SCS-4	Sharp	vcon-scs4.cmp	S	S	S											
	SCS-4_e	Sharp	vcon-scs4_e.cmp				S	S									
	SCS-5	Sharp	vcon-scs5.cmp	S	S	S											
	SCS-6_e	Sharp	vcon-scs5_e.cmp				S	S									
	SCS-6	Sharp	vcon-scs6.cmp	S	S	S											
	SCS-6_e	Sharp	vcon-scs6_e.cmp				S	S									
	SCS-7	Sharp	vcon-scs7.bits	S	S	S											
	SCS-7_e	Sharp	vcon-scs7_e.bits				S	S									
	SCS-8	Sony	vcon-scs8.bits	D													
	SCS-8_e	Sony	vcon-scs8_e.bits				D										
	SCS-9	Sony	vcon-scs9.bits		D												
	SCS-9_e	Sony	vcon-scs9_e.bits					D									
	SCS-10	Sharp	vcon-scs10.cmp	D													
	SCS-10_e	Sharp	vcon-scs10_e.cmp				D										
	SCS-11	Sharp	vcon-scs11.cmp		D												
	SCS-11_e	Sharp	vcon-scs11_e.cmp					D									
Scalability	SCC-1	Sharp	vcon-scc1.cmp						S	S							
	SCC-1_e	Sharp	vcon-scc1_e.cmp						S	S							
	SCC-2	Sharp	vcon-scc2.cmp						S	S							
	SCC-2_e	Sharp	vcon-scc2_e.cmp						S	S							
	SCC-3	Sharp	vcon-scc3.cmp						S	S							
	SCC-3_e	Sharp	vcon-scc3_e.cmp						S	S							
	SCC-4	Sharp	vcon-scc4.cmp						S	S							
	SCC-4_e	Sharp	vcon-scc4_e.cmp						S	S							
	SCC-5	Sharp	vcon-scc5.cmp						S	S							
	SCC-5_e	Sharp	vcon-scc5_e.cmp						S	S							
	SCC-6	Sharp	vcon-scc6.cmp						S	S							
	SCC-6_e	Sharp	vcon-scc6_e.cmp						S	S							
	SCC-7	Sharp	vcon-scc7.cmp						D								
	SCC-7_e	Sharp	vcon-scc7_e.cmp						D								
	SCC-8	Sharp	vcon-scc8.cmp							D							
	SCC-8_e	Sharp	vcon-scc8_e.cmp							D							
Error Resilience	er-1	Toshiba	Vcon-er1.cmp			S											
	er-2-1	Toshiba	Vcon-er2-1.cmp	S													
	er-2-2	Toshiba	Vcon-er2-2.cmp		S												
	er-2-3	Toshiba	Vcon-er2-3.cmp			S											
	er-3-1	Toshiba	Vcon-er3-1.cmp	S													
	er-3-2	Toshiba	Vcon-er3-2.cmp		S												
	er-3-3	Toshiba	Vcon-er3-3.cmp			S											

Categories	Bitstream	Donated by	Bitstreams Name	Simple			Simple Sclable (Enhance)		Core		Main			N-Bit	Scalable Texture		
				L1	L2	L3	L1	L2	L1	L2	L2	L3	L4	L2	L1	L2	L3
Scalable Still Texture	ss-1	Sharp	vcon-ss1.bits								S	S	S		S	S	
	ss-2	Sharp	vcon-ss2.bits								S	S	S		S	S	
	ss-3	Sharp	vcon-ss3.bits								S	S	S		S	S	
	ss-4	Sharp	vcon-ss4.bits								S	S	S		S	S	
	ss-5	Sharp	vcon-ss5.bits								S	S	S		S	S	
	ss-6	Sharp	vcon-ss6.bits								S	S	S		S	S	
	ss-7	Sharp	vcon-ss7.bits								S	S	S		S	S	
	ss-8	Sarnoff	vcon-ss8.bits								S	S	S		S	S	
	ss-9	Sarnoff	vcon-ss9.bits								S	S	S		S	S	
	ss-10	Sarnoff	vcon-ss10.bits								S	S	S		S	S	
	ss-11	Sarnoff	vcon-ss11.bits								S	S	S		S	S	
	ss-12	TI	vcon-ss12.bits								S	S	S		S	S	
	ss-13	TI	vcon-ss13.bits								S	S	S		S	S	
Sprites	sp1	Hitachi	vcon-sp1.bits								X						
	sp2	Hitachi	vcon-sp2.bits									X					
	sp3	Hitachi	vcon-sp3.bits										X				
	sp4	Hitachi	vcon-sp4.bits								X						
	sp5	Hitachi	vcon-sp5.bits									X					
	sp6	Hughes	vcon-sp6.bits									X					

5.5.8 Bitstream Donated by MPEG-4 Platform Verification Bitstream Development Project

5.5.8.1 Simple Profile bitstreams

The list of the bitstreams donated by the MPEG-4 Platform Verification Bitstream Development Project of Japan is provided in this clause.

Table 11 — I-VOP verification bitstream suite

File Name	Profile@Level	Test Sequence	Duration of Sequence [s]	Bit Rate [kbit/s]	Image Size (horizontal) [pel]	Image Size (vertical) [pel]	Number of Coded VOPs	Bitstream Specifications
hit000.m4v	Simple@L3	Octopus	1.667	384	352	288	1	basic
jvc000.m4v	Simple@L3	Friends	10.000	384	176	144	100	basic
mit000.m4v	Simple@L2	Aki	0.500	128	352	288	5	basic
mit001.m4v	Simple@L1	Talk	1.000	64	176	144	8	AC/DC prediction
mit002.m4v	Simple@L1	Talk	1.000	64	176	144	8	quantisation
mit003.m4v	Simple@L1	Talk	1.000	64	176	144	8	intra_vlc_thr
mit004.m4v	Simple@L1	Maiko	10.000	64	176	144	30	VBV(L1)
mit005.m4v	Simple@L2	Aki	10.000	128	352	288	75	VBV(L2)
mit006.m4v	Simple@L3	Maiko	10.000	384	352	288	50	VBV(L3)
san000.m4v	Simple@L2	Aki1	10.000	64	352	288	50	basic
san001.m4v	Simple@L2	Aki1	10.000	128	352	288	100	AC prediction

Table 12 — P-VOP verification bitstream suite

File Name	Profile@Level	Test Sequence	Duration of Sequence [s]	Bit Rate [kbit/s]	Image Size (horizontal) [pel]	Image Size (vertical) [pel]	Number of Coded VOPs	Bitstream Specifications
hit001.m4v	Simple@L3	Talk	10.000	256	352	288	100	basic
hit002.m4v	Simple@L3	Aki1	10.000	96	352	288	100	escape code type 1
hit003.m4v	Simple@L3	Maiko	10.000	384	352	288	100	escape code type 2
hit004.m4v	Simple@L3	Drive	10.000	256	352	288	100	escape code type 3
hit005.m4v	Simple@L3	Talk	10.000	180	352	288	100	dquant
hit006.m4v	Simple@L3	Aki1	10.000	72	352	288	100	intra_dc_vlc_thr
hit007.m4v	Simple@L3	Maiko	10.000	384	352	288	100	AC prediction
hit008.m4v	Simple@L3	Drive	10.000	240	352	288	100	vop_rounding_type
hit009.m4v	Simple@L3	Friends	10.000	360	352	288	100	vop_fcode_forward
hit010.m4v	Simple@L3	Maiko	10.000	384	352	288	100	unrestricted MV
hit011.m4v	Simple@L3	Talk	10.000	256	352	288	100	4MV
hit012.m4v	Simple@L3	Aki1	10.000	64	352	288	100	vop_coded
hit013.m4v	Simple@L3	Octopus	10.000	80	352	288	9	modulo_time_base, vop_time_increment
hit014.m4v	Simple@L3	Drive	10.000	280	352	288	100	GOV header
jvc001.m4v	Simple@L3	Talk	10.000	384	352	288	150	basic
jvc002.m4v	Simple@L3	Talk	10.000	384	352	288	150	GOV
jvc003.m4v	Simple@L1	Aki1	10.000	64	80	144	150	VBV (L1)
jvc004.m4v	Simple@L2	Aki1	10.000	128	176	288	150	VBV (L2)
jvc005.m4v	Simple@L3	Aki1	10.000	384	176	288	300	VBV (L3)
jvc006.m4v	Simple@L1	Aki1	10.000	64	80	144	300	VCV (L1)
jvc007.m4v	Simple@L2	Aki1	10.000	128	176	288	300	VCV (L2)
jvc008.m4v	Simple@L3	Aki1	10.000	384	176	288	600	VCV (L3)
jvc009.m4v	Simple@L1	Aki1	10.000	64	176	144	150	VMV (L1)
jvc010.m4v	Simple@L2	Aki1	10.000	128	352	288	150	VMV (L2)
jvc011.m4v	Simple@L3	Aki1	10.000	384	352	288	300	VMV (L3)
jvc012.m4v	Simple@L3	Drive	10.000	384	352	288	150	quant-dquant
jvc013.m4v	Simple@L3	Drive	10.000	384	352	288	150	quant-intra_dc_vlc_thr
jvc014.m4v	Simple@L3	Maiko	10.000	150	352	288	150	No MC
jvc015.m4v	Simple@L3	Own synthetic	10.000	48	352	288	150	2 pel MC
jvc016.m4v	Simple@L3	Own synthetic	10.000	384	352	288	150	1 pel MC
jvc017.m4v	Simple@L3	Talk	10.000	384	352	288	150	0.5pel MC
jvc018.m4v	Simple@L3	Talk	10.000	384	352	288	150	4MV
jvc019.m4v	Simple@L3	Talk	10.000	384	352	288	150	unrestricted MC
jvc020.m4v	Simple@L3	Talk	10.000	384	352	288	150	vop_rounding_type
jvc021.m4v	Simple@L3	Talk	10.000	384	352	288	150	f_code
mit007.m4v	Simple@L1	Talk	10.000	64	176	144	150	basic
mit008.m4v	Simple@L2	Talk	10.000	128	352	288	150	f_code
mit009.m4v	Simple@L2	Aki	10.000	128	352	288	150	4MV
mit010.m4v	Simple@L2	Talk	10.000	128	352	288	150	vop_rounding_type
mit011.m4v	Simple@L2	Talk	10.000	128	352	288	150	unrestricted MC
mit012.m4v	Simple@L1	Talk	10.000	64	176	144	104	VBV(L1)
mit013.m4v	Simple@L2	Talk	10.000	128	352	288	150	VBV(L2)
mit014.m4v	Simple@L3	Talk	10.000	384	352	288	300	VBV(L3)
mit015.m4v	Simple@L1	Talk	10.000	64	176	144	70	frame drop
mit016.m4v	Simple@L1	own synthetic	1.000	64	528	48	10	input format(L1)

File Name	Profile@Level	Test Sequence	Duration of Sequence [s]	Bit Rate [kbit/s]	Image Size (horizontal) [pel]	Image Size (vertical) [pel]	Number of Coded VOPs	Bitstream Specifications
mit017.m4v	Simple@L2	own synthetic	1.000	128	288	352	5	input format(L2)
mit018.m4v	Simple@L3	own synthetic	1.000	384	704	144	15	input format(L3)
mit019.m4v	Simple@L2	Aki	5.000	128	352	288	74	GOV
san002.m4v	Simple@L2	Aki1	10.000	128	352	288	136	basic
san003.m4v	Simple@L2	Aki1	10.000	128	352	288	136	GOV
san004.m4v	Simple@L1	Aki1	10.000	64	176	144	80	VBV (L1)
san005.m4v	Simple@L2	Aki1	10.000	128	352	288	58	VBV (L2)
san006.m4v	Simple@L3	Aki1	10.000	384	352	288	173	VBV (L3)
san007.m4v	Simple@L1	Aki1	10.000	64	176	144	134	VCV (L1)
san008.m4v	Simple@L2	Aki1	10.000	128	352	288	58	VCV (L2)
san009.m4v	Simple@L3	Aki1	10.000	384	352	288	173	VCV (L3)
san010.m4v	Simple@L1	Talk	10.000	64	176	144	90	VMV (L1)
san011.m4v	Simple@L2	Talk	10.000	128	352	288	88	VMV (L2)
san012.m4v	Simple@L3	Talk	10.000	384	352	288	100	VMV (L3)
san013.m4v	Simple@L2	Aki1	10.000	128	352	288	150	dquant
san014.m4v	Simple@L2	Aki1	10.000	128	352	288	149	intra_dc_vlc_thr
san015.m4v	Simple@L2	Aki1	10.000	128	352	288	149	2 pel MC
san016.m4v	Simple@L2	Aki1	10.000	128	352	288	149	1 pel MC
san017.m4v	Simple@L2	Aki1	10.000	128	352	288	150	f_code
san018.m4v	Simple@L2	Aki1	10.000	128	352	288	150	vop_rounding_type
san019.m4v	Simple@L2	Aki1	10.000	128	352	288	149	4MV
san020.m4v	Simple@L2	Talk	10.000	128	352	288	149	unrestricted MC

Table 13 — Error resilience verification bitstream suite

File Name	Profile@Level	Test Sequence	Duration of Sequence [s]	Bit Rate [kbit/s]	Image Size (horizontal) [pel]	Image Size (vertical) [pel]	Number of Coded VOPs	Bitstream Specifications
hit025.m4v	Simple@L1	Aki1	10.000	64	176	144	100	resync_marker
hit026.m4v	Simple@L1	Aki1	10.000	64	176	144	100	HEC
hit027.m4v	Simple@L1	Friends	10.000	64	176	144	50	data partitioning (I-VOP)
hit028.m4v	Simple@L1	Drive	10.000	64	176	144	100	data partitioning (P-VOP)
hit029.m4v	Simple@L1	Talk	10.000	64	176	144	100	reversible VLC
hit030.m4v	Simple@L1	Drive	10.000	64	176	144	100	escape code (RVLC)
mit025.m4v	Simple@L2	Talk	10.000	128	352	288	150	video packet-packet length
mit026.m4v	Simple@L2	Talk	10.000	128	352	288	150	video packet-HEC
mit027.m4v	Simple@L1	Talk	1.000	64	176	144	8	data partitioning-I-VOP
mit028.m4v	Simple@L1	Talk	10.000	64	176	144	150	data partitioning-P-VOP
mit029.m4v	Simple@L1	Friends	2.000	64	176	144	6	reversible VLC

Table 14 — Short header mode verification bitstream suite

File Name	Profile@Level	Test Sequence	Duration of Sequence [s]	Bit Rate [kbit/s]	Image Size (horizontal) [pel]	Image Size (vertical) [pel]	Number of Coded VOPs	Bitstream Specifications
hit031.m4v	Simple@L3	Aki1	1.667	384	352	288	1	I-VOP
hit032.m4v	Simple@L3	Aki1	10.000	35	352	288	100	P-VOP
hit033.m4v	Simple@L3	Drive	10.000	281	352	288	100	escape code
hit034.m4v	Simple@L3	Talk	10.000	180	352	288	100	dquant
hit035.m4v	Simple@L3	Aki1	10.000	45	352	288	100	GOB header
hit036.m4v	Simple@L3	Aki1	10.000	41	352	288	100	user data
hit037.m4v	Simple@L3	Talk	10.000	272	352	288	100	MB stuffing
hit038.m4v	Simple@L1	Drive	10.000	64	176	144	150	VBV (L1)
hit039.m4v	Simple@L2	Talk	10.000	128	352	288	150	VBV (L2)
hit040.m4v	Simple@L3	Drive	10.000	384	352	288	300	VBV (L3)
jvc022.m4v	Simple@L3	Octopus	10.000	384	176	144	100	basic
jvc023.m4v	Simple@L1	Talk	10.000	64	176	144	100	VBV (L1)
jvc024.m4v	Simple@L2	Talk	10.000	128	352	288	100	VBV(L2)
jvc025.m4v	Simple@L3	Talk	10.000	384	352	288	150	VBV (L3)
mit020.m4v	Simple@L1	Talk	5.03	55	176	144	73	basic
mit021.m4v	Simple@L1	Talk	5.07	55	176	144	72	VBV(L1)
mit022.m4v	Simple@L2	Drive	5.03	119	352	288	67	VBV(L2)
mit023.m4v	Simple@L3	Talk	4.9	121	352	288	144	VBV(L3)
mit024.m4v	Simple@L2	Drive	5.03	128	352	288	67	GOB
san021.m4v	Simple@L2	Aki1	10.000	128	352	288	99	basic
san022.m4v	Simple@L1	Aki1	10.000	64	176	144	100	VBV (L1)
san023.m4v	Simple@L2	Aki1	10.000	128	352	288	49	VBV (L2)
san024.m4v	Simple@L3	Aki1	10.000	384	352	288	127	VBV (L3)

Table 15 — Overall verification bitstream suite

File Name	Profile@Level	Test Sequence	Duration of Sequence [s]	Bit Rate [kbit/s]	Image Size (horizontal) [pel]	Image Size (vertical) [pel]	Number of Coded VOPs	Bitstream Specifications
hit016.m4v	Simple@L1	Drive	10.000	64	176	144	150	VBV (L1)
hit017.m4v	Simple@L2	Aki1	10.000	128	352	288	150	VBV (L2)
hit018.m4v	Simple@L3	Talk	10.000	384	352	288	300	VBV (L3)
hit019.m4v	Simple@L1	Drive	10.000	64	176	144	150	VMV (L1)
hit020.m4v	Simple@L2	Talk	10.000	128	352	288	150	VMV (L2)
hit021.m4v	Simple@L3	Drive	10.000	384	352	288	300	VMV (L3)
hit022.m4v	Simple@L1	Aki1	10.000	64	176	144	150	VCV (L1)
hit023.m4v	Simple@L2	Aki1	10.000	128	352	288	150	VCV (L2)
hit024.m4v	Simple@L3	Aki1	10.000	384	352	288	300	VCV (L3)
mit030.m4v	Simple@L1	Talk	3.000	64	176	144	45	MB stuffing
mit031.m4v	Simple@L1	Talk	10.000	64	176	144	150	all P-VOP coding

5.5.8.2 Core Profile bitstreams

Table 16 — I-VOP Verification Bitstream suite

File Name	Profile@Level	Test Sequence	Duration of Sequence [s]	Bit Rate [kbit/s]	Image Size (horizontal)[pel]	Image Size (vertical)[pel]	Number of Coded VOPs	Bitstream Specifications
mat000.m4v	Core@L1	own synthetic	66.600	116	16	16	999	IVOP IDCT bitstream1
mat001.m4v	Simple@L1	own synthetic	66.600	30	16	16	999	IVOP IDCT bitstream2
mat002.m4v	Core@L1	own synthetic	6.570	384	176	144	25	IVOP VBV core@L1
mat003.m4v	Core@L2	own synthetic	6.549	2000	352	288	25	IVOP VBV core@L2
mat004.m4v	Simple@L1	own synthetic	29.515	64	176	144	27	IVOP VBV simple@L1
mat005.m4v	Simple@L2	own synthetic	61.584	128	352	288	25	IVOP VBV simple@L2
mat006.m4v	Simple@L3	own synthetic	20.528	384	352	288	25	IVOP VBV simple@L3
mat007.m4v	Simple@L3	Stefan	0.033	384	128	16	1	IVOP Table B-06 VLCs
mat008.m4v	Simple@L2	Gold fish	0.085	128	256	16	1	IVOP Table B-08 (intra) VLCs
mat009.m4v	Simple@L1	Gold fish	0.293	64	144	16	1	IVOP Table B-13 VLCs
mat010.m4v	Simple@L1	Gold fish	0.300	64	160	16	1	IVOP Table B-14 VLCs
mat011.m4v	Core@L1	Stefan	0.033	384	352	16	1	IVOP Table B-16 +ve VLCs
mat012.m4v	Core@L1	Stefan	0.033	384	352	16	1	IVOP Table B-16 -ve VLCs
mat013.m4v	Simple@L1	Stefan	0.971	64	352	32	1	IVOP Table B-19 +ve VLCs
mat014.m4v	Simple@L1	Stefan	0.971	64	352	32	1	IVOP Table B-19 -ve VLCs
mat015.m4v	Simple@L1	Stefan	0.602	64	352	32	1	IVOP Table B-21 +ve VLCs
mat016.m4v	Simple@L1	Stefan	0.602	64	352	32	1	IVOP Table B-21 -ve VLCs
nec000.m4v	Core@L1	aki1	10.000	384	176	144	300	I-VOP(H.263 Quantization)
nec001.m4v	Core@L2	talk	10.000	2000	352	288	300	I-VOP MPEG Quantization)

Table 17 — P-VOP Verification Bitstream suite

File Name	Profile@Level	Test Sequence	Duration of Sequence [s]	Bit Rate [kbit/s]	Image Size(horizontal) [pel]	Image Size (vertical)[pel]	Number of Coded VOPs	Bitstream Specifications
mat017.m4v	Core@L2	sax.cif	0.333	2000	352	288	10	PVOP 1 motion vector
mat018.m4v	Simple@L1	akiyo.qcif	12.023	64	176	144	41	PVOP 4 motion vector
mat019.m4v	Core@L1	sax.cif	0.067	139	16	16	2	PVOP saturation
mat020.m4v	Simple@L1	own synthetic	9.941	1	16	16	169	PVOP Table B-12 VLCs
mat021.m4v	Core@L1	own synthetic	2.059	384	176	144	9	PVOP VBV core@L1
mat022.m4v	Core@L2	own synthetic	2.037	2000	352	288	11	PVOP VBV core@L2
mat023.m4v	Simple@L1	own synthetic	9.943	64	176	144	9	PVOP VBV simple@L1
mat024.m4v	Simple@L2	own synthetic	22.114	128	352	288	10	PVOP VBV simple@L2
mat025.m4v	Simple@L3	own synthetic	7.371	384	352	288	10	PVOP VBV simple@L3
mat026.m4v	Simple@L2	Stefan	2.286	64	352	32	2	PVOP Table B-07 VLCs
mat027.m4v	Simple@L1	Stefan	0.475	64	256	16	2	PVOP Table B-08 (inter) VLCs

File Name	Profile@Level	Test Sequence	Duration of Sequence [s]	Bit Rate [kbit/s]	Image Size(horizontal) [pel]	Image Size (vertical)[pel]	Number of Coded VOPs	Bitstream Specifications
mat028.m4v	Simple@L1	Gold fish	0.211	64	176	144	3	PVOP Table B-17 +ve VLCs
mat029.m4v	Simple@L1	Gold fish	0.211	64	176	144	3	PVOP Table B-17 -ve VLCs
mat030.m4v	Simple@L2	Stefan	1.156	64	352	32	2	PVOP Table B-20 +ve VLCs
mat031.m4v	Simple@L2	Stefan	1.156	64	352	32	2	PVOP Table B-20 -ve VLCs
mat032.m4v	Core@L1	Stefan	0.667	108	352	32	2	PVOP Table B-22 +ve VLCs
mat033.m4v	Core@L1	Stefan	0.667	108	352	32	2	PVOP Table B-22 -ve VLCs
nec002.m4v	Core@L1	maiko	10.000	384	176	144	300	P-VOP(H.263 Quantization)
nec003.m4v	Core@L2	drive	10.000	2000	352	288	300	P-VOP(MPEG Quantization)
nec006.m4v	Core@L1	octpus	10.000	384	176	144	300	P-VOP 4MV (H.263 Quantization)
nec007.m4v	Core@L2	maiko	10.000	2000	352	288	300	P-VOP 4MV (MPEG Quantization)

Table 18 — B-VOP Verification Bitstream suite

File Name	Profile@Level	Test Sequence	Duration of Sequence [s]	Bit Rate [kbit/s]	Image Size (horizontal)[pel]	Image Size (vertical)[pel]	Number of Coded VOPs	Bitstream Specifications
mat034.m4v	Core@L1	Gold fish	55.455	384	176	144	61	BVOP forward MV
mat035.m4v	Core@L1	Gold fish	55.455	384	176	144	61	BVOP backward MV
mat036.m4v	Core@L1	Gold fish	55.455	384	176	144	61	BVOP bi-directional MV
mat037.m4v	Core@L1	Gold fish	52.727	384	176	144	58	BVOP direct MV
mat038.m4v	Core@L1	Gold fish	5.455	384	176	144	6	BVOP Table B-3 VLCs
mat039.m4v	Core@L1	Gold fish	10.000	384	176	144	11	BVOP Table B-4 VLCs
mat040.m4v	Core@L1	own synthetic	1.948	384	176	144	9	BVOP VBV core@L1
mat041.m4v	Core@L2	own synthetic	1.714	2000	352	288	9	BVOP VBV core@L2
nec010.m4v	Core@L1	friends	10.000	384	176	144	298	B-VOP (Fwd, Bwd, Interpolation)
nec011.m4v	Core@L2	drive	10.000	2000	352	288	298	B-VOP (Fwd, Bwd, Interpolation)
nec012.m4v	Core@L1	maiko	10.000	384	176	144	298	B-VOP (Direct Mode)
nec013.m4v	Core@L2	octpus	10.000	2000	352	288	298	B-VOP (Direct Mode)

Table 19 — AC/DC Prediction Verification Bitstream suite

File Name	Profile@Level	Test Sequence	Duration of Sequence [s]	Bit Rate [kbit/s]	Image Size(horizontal) [pel]	Image Size (vertical)[pel]	Number of Coded VOPs	Bitstream Specifications
mat042.m4v	Simple@L2	own synthetic	0.588	16	32	32	10	AC/DC prediction (Intra)
mat043.m4v	Simple@L2	own synthetic	0.588	5	32	32	10	AC/DC prediction (Inter)
mat044.m4v	Simple@L3	own synthetic	0.588	10	32	32	10	AC/DC ac_pred_flag
mat045.m4v	Simple@L1	own synthetic	0.118	64	16	16	2	AC/DC Saturation
mat046.m4v	Core@L2	own synthetic	0.118	178	64	64	2	AC/DC @ Shape Boundary
nec004.m4v	Core@L1	friends	10.000	384	176	144	300	AC prediction
nec005.m4v	Core@L2	octpus	10.000	2000	352	288	300	AC prediction

Table 20 — Quantization Method Verification Bitstream suite

File Name	Profile@Level	Test Sequence	Duration of Sequence [s]	Bit Rate [kbit/s]	Image Size (horizontal)[pel]	Image Size (vertical)[pel]	Number of Coded VOPs	Bitstream Specifications
mat047.m4v	Simple@L2	akiyo	0.333	117	176	144	10	Quantization method 1(I-VOP & P-VOP)
mat048.m4v	Simple@L2	akiyo	0.333	117	176	144	10	Quantization method 2(I-VOP & P-VOP)
mat049.m4v	Core@L2	akiyo	0.367	122	176	144	11	Quantization method 1(B-VOP)
mat050.m4v	Core@L2	akiyo	0.367	112	176	144	11	Quantization method 2(B-VOP)
mat051.m4v	Simple@L3	akiyo	0.333	180	176	144	10	Quantization DC Scaler
mat052.m4v	Simple@L3	akiyo	7.333	132	176	144	11	Quantization Matrix Intra
mat053.m4v	Core@L2	akiyo	0.333	612	176	144	10	Quantization Matrix Inter
mat054.m4v	Core@L1	own synthetic	0.067	189	16	16	2	Quantization Saturation
nec014.m4v	Core@L1	friends	10.000	384	176	144	298	Quantization Matrix Intra/Inter
nec015.m4v	Core@L2	octpus	10.000	2000	352	288	298	Quantization Matrix Intra/Inter
pio000.m4v	Core@L1	octpus	10.000	384	176	144	300	Variable Q + intra_dc_vlc_thr
pio001.m4v	Core@L1	maiko	10.000	384	176	144	300	Variable Q + Load WQ Mtrx
pio002.m4v	Core@L1	friends	10.000	384	176	144	300	Video Packet + Variable Q
pio003.m4v	Core@L1	drive	10.000	384	176	144	300	Data partitioning + Variable Q
pio004.m4v	Core@L1	octpus	10.000	384	176	144	300	RVLC + Variable Q

Table 21 — Binary Shape Verification Bitstream suite

File Name	Profile@Level	Test Sequence	Duration of Sequence [s]	Bit Rate [kbit/s]	Image Size (horizontal)[pel]	Image Size (vertical)[pel]	Number of Coded VOPs	Bitstream Specifications
mat055.m4v	Core@L2	Claire_qcif	3.118	44	176	144	53	BSVOP shape motion vector
mat056.m4v	Core@L1	own synthetic	0.118	72	64	64	2	BSVOP shape motion vector predictor MVs1,2,3
mat057.m4v	Core@L1	own synthetic	0.118	384	64	64	2	BSVOP shape motion vector predictor MVs4,5,6
mat058.m4v	Core@L1	Gold fish	0.014	384	16	16	8	BSVOP Table B-09 (INTRA)
mat059.m4v	Core@L1	Gold fish	0.046	384	16	16	9	BSVOP Table B-09 (INTER)
mat060.m4v	Core@L1	Gold fish	0.010	384	16	16	4	BSVOP Table B-10 (INTRA)
mat061.m4v	Core@L1	Gold fish	0.019	384	16	16	5	BSVOP Table B-10 (INTER)
mat062.m4v	Core@L1	Gold fish	0.004	384	16	16	2	BSVOP Table B-11 (INTRA)
mat063.m4v	Core@L1	Gold fish	0.012	384	16	16	3	BSVOP Table B-11 (INTER)
mat064.m4v	Core@L1	Aki2_qcif	0.833	384	176	144	25	BSVOP Table B-27
mat065.m4v	Core@L2	sax_cif	1.633	2000	352	288	49	BSVOP Table B-28
mat066.m4v	Core@L2	Bike_qcif	1.467	2000	176	144	22	BSVOP Table B-29
mat067.m4v	Core@L1	Pose_qcif	1.600	384	176	144	8	BSVOP Table B-30
mat068.m4v	Core@L1	Goldfish	2.000	384	176	144	10	BSVOP Table B-32 (INTRA)
mat069.m4v	Core@L1	Akiyo_qcif	2.667	384	16	16	80	BSVOP Table B-32 (INTER)
mat070.m4v	Core@L1	own synthetic	2.423	384	176	144	14	BSVOP VBV core@L1
mat071.m4v	Core@L2	own synthetic	3.191	2000	352	288	15	BSVOP VBV core@L2
nec020.m4v	Core@L1	goldfish	10.000		176	144	300	Binary Shape Only
nec021.m4v	Core@L2	bike	10.000		352	288	300	Binary Shape Only
nec022.m4v	Core@L1	goldfish	10.000	384	176	144	300	Binary Shape (I,P-VOP)
nec023.m4v	Core@L2	bike	10.000	2000	352	288	300	Binary Shape (I,P-VOP)

Table 22 — Error Resilience Verification Bitstream suite

File Name	Profile@Level	Test Sequence	Duration of Sequence [s]	Bit Rate [kbit/s]	Image Size(horizontal)[pel]	Image Size (vertical)[pel]	Number of Coded VOPs	Bitstream Specifications
nec016.m4v	Core@L1	octpus	10.000	384	176	144	300	Error Resilience(VP+DP)
nec017.m4v	Core@L2	octpus	10.000	2000	352	288	300	Error Resilience (VP+DP)
nec018.m4v	Core@L1	octpus	10.000	384	176	144	300	Error Resilience (VP+DP+RVLC)
nec019.m4v	Core@L2	octpus	10.000	2000	352	288	300	Error Resilience (VP+DP+RVLC)
pio005.m4v	Core@L1	aki2	10.000	192	176	144	300	Bianry Shape (Variable Q + intra_dc_vlc_thr)
pio006.m4v	Core@L1	bike	10.000	384	176	144	300	Bianry Shape (Video Packet + Variable Q)
pio007.m4v	Core@L1	goldfish	10.000	384	176	144	300	Bianry Shape (Data partitioning + Variable Q)
pio008.m4v	Core@L1	goldfish	10.000	384	176	144	300	Bianry Shape (RVLC + Variable Q)

Table 23 — The Short Header Verification Bitstream suite

File Name	Profile@Level	Test Sequence	Duration of Sequence [s]	Target Bit Rate [kbit/s]	Image Size(horizontal)[pel]	Image Size (vertical)[pel]	Number of Coded VOPs	Bitstream Specifications
nec008.m4v	Core@L1	drive	10.000	384	176	144	299	Short Header
nec009.m4v	Core@L2	friends	10.000	2000	352	288	300	Short Header

Table 24 — The Overall Test Bitstream suite

File Name	Profile@Level	Test Sequence	Duration of Sequence [s]	Target Bit Rate [kbit/s]	Image Size(horizontal)[pel]	Image Size (vertical)[pel]	Number of Coded VOPs	Bitstream Specifications
pio009.m4v	Core@L2	maiko	10.000	2000	352	288	300	Conformance (Core_L2_00)
pio010.m4v	Core@L2	friends	10.000	2000	352	288	300	Conformance (Core_L2_01)
pio011.m4v	Core@L2	goldfish	10.000	2000	352	288	300	Conformance (Core_L2_02)
pio012.m4v	Core@L2	goldfish	10.000	2000	352	288	300	Conformance (Core_L2_02)

5.6 Additional Conformance Testing

5.6.1 Specification of the test bitstreams

5.6.1.1 Test Bitstreams - General

5.6.1.1.1 Test bitstream #A1GE-1

Specification: A series of consecutive B-VOPs with all macroblocks using bi-directional interlaced prediction. Number of MB/s and bitrate are the maximum allowed for the profile-and-level combination. Quarter-sample interpolation in both the horizontal and vertical directions, for all luminance and chrominance blocks.

Functional stage: prediction bandwidth

Purpose: Check that the decoder handles the worst case of prediction bandwidth including quarter-sample interpolation. Reference VOP buffers organised progressively (interleaved fields) and macroblocks stored in contiguous address page segments would have the greatest penalty. Effective filtered block size is 16x8 for luminance and 8x4 for chrominance.

5.6.1.1.2 Test bitstream #A1GE-2

Specification: A series of consecutive interlaced coded P-VOPs with all macroblocks using both top and bottom field of the reference VOP. Number of MB/s and bitrate are the maximum allowed for the profile-and-level combination. Maximize number of quarter-sample prediction in both the horizontal and vertical directions, for both luminance and chrominance blocks.

Functional stage: prediction bandwidth

Purpose: Check that the decoder handles the worst case of prediction bandwidth including quarter-sample interpolation. Prediction bandwidth is at a maximum in this mode due to the small block sizes and two prediction sources.

5.6.1.1.3 Test bitstream #A1GE-3

Specification: A series of consecutive interlaced coded S(GMC)-VOPs. As many affine warping transformation(no_of_sprite_warping_points==3) with 1/16 pixel accuracy (sprite_warping_accuracy==3) as possible, luminance and chrominance blocks. Number of MB/s and bitrate are the maximum allowed for the profile-and-level combination. Maximize number of half-sample prediction in both the horizontal and vertical directions, for both luminance and chrominance blocks.

Functional stage: prediction bandwidth

Purpose: Check that the decoder handles the worst case of prediction bandwidth. Prediction bandwidth is at a maximum in this mode due to the small block sizes and two prediction sources.

5.6.1.1.4 Test bitstream #A1GE-4

Specification: Bitstream with only intra macroblocks using only the DC coefficient and predicted macroblocks having no DCT coefficients. Reconstructed motion vectors used for predicting both luminance and chrominance have all possible combinations of quarter-sample, half-sample and full-sample values, both for the horizontal and the vertical coordinates, and all those combinations are used for each prediction mode in both progressive and interlaced coded VOPs.

Functional stage: MCP

Purpose: Check that decoder implements motion compensation stages with full accuracy in all cases including quarter-sample interpolation. Except for reconstruction of Intra DC blocks, the test does not involve other decoder functions such as IDCT, inverse quantization and mismatch control. When a static decoder test is performed using the static test technique described in this document, the decoder under test shall reconstruct samples identical to those reconstructed by a reference decoder for all predicted macroblocks.

5.6.1.1.5 Test bitstream #A1GE-5

Specification: Bitstream with only intra macroblocks using only the DC coefficient and predicted macroblocks having no DCT coefficients. GMC macroblocks are included in predicted macroblocks. Reconstructed motion vectors of non GMC macroblocks used for predicting both luminance and chrominance have all possible combinations of half-sample and full-sample values, both for the horizontal and the vertical coordinates. Translational warping (no_of_sprite_warping_points==1) with 1/2 pixel accuracy(sprite_warping_accuracy==0) is used for reconstructing both luminance and chrominance samples in GMC macroblocks. All those combinations are used for each prediction mode in progressive coded VOPs including S(GMC)-VOPs.

Functional stage: MCP

Purpose: Check that decoder implements motion compensation stages with full accuracy, and translational warping stage with 1/2 pixel accuracy. And check for implementation motion vector decoding stages as defined in subclause 7.8.7.3 of ISO/IEC 14496-2/Amd-1). Except for reconstruction of Intra DC blocks, the test does not involve other decoder functions such as IDCT, inverse quantization and mismatch control. When a static decoder test is performed using the static test technique described in this document, the decoder under test shall reconstruct samples identical to those reconstructed by a reference decoder for all predicted macroblocks.

5.6.1.1.6 Test bitstream #A1GE-6

Specification: Bitstream with only intra macroblocks using only the DC coefficient and predicted macroblocks having no DCT coefficients. GMC macroblocks are included in predicted macroblocks. Reconstructed motion vectors of non GMC macroblocks used for predicting both luminance and chrominance have all possible combinations of half-sample and full-sample values, both for the horizontal and the vertical coordinates. Isotropic warping(no_of_sprite_warping_points==2) with 1/4 pixel accuracy(sprite_warping_accuracy==1) is used for reconstructing both luminance and chrominance samples in GMC macroblocks. All those combinations are used for each prediction mode in progressive coded VOPs. Including S(GMC)-VOPs

Functional stage: MCP

Purpose: Check that decoder implements motion compensation stages with full accuracy, and isotropic warping stage with 1/4 pixel accuracy. And check for implementation motion vector decoding stages as defined in subclause 7.8.7.3 of ISO/IEC 14496-2/Amd-1). Except for reconstruction of Intra DC blocks, the test does not involve other decoder functions such as IDCT, inverse quantization and mismatch control. When a static decoder test is performed using the static test technique described in this document, the decoder under test shall reconstruct samples identical to those reconstructed by a reference decoder for all predicted macroblocks.

5.6.1.1.7 Test bitstream #A1GE-7

Specification: Bitstream with only intra macroblocks using only the DC coefficient and predicted macroblocks having no DCT coefficients. GMC macroblocks are included in predicted macroblocks. Reconstructed motion vectors of non GMC macroblocks used for predicting both luminance and chrominance have all possible combinations of half-sample and full-sample values, both for the horizontal and the vertical coordinates. Affine

warping (no_of_sprite_warping_points==3) with 1/8 pixel accuracy(sprite_warping_accuracy==2) is used for reconstructing both luminance and chrominance samples in GMC macroblocks. All those combinations are used for each prediction mode in both progressive and interlaced coded VOPs including S(GMC)-VOPs.

Functional stage: MCP

Purpose: Check that decoder implements motion compensation stages with full accuracy, and affine warping stage with 1/8 pixel accuracy And check for implementation motion vector decoding stages as defined in subclause 7.8.7.3 of ISO/IEC 14496-2/Amd-1). Except for reconstruction of Intra DC blocks, the test does not involve other decoder functions such as IDCT, inverse quantization and mismatch control. When a static decoder test is performed using the static test technique described in this document, the decoder under test shall reconstruct samples identical to those reconstructed by a reference decoder for all predicted macroblocks.

5.6.1.1.8 Test bitstream #A1GE-8

Specification: Bitstream with only intra macroblocks using only the DC coefficient and predicted macroblocks having no DCT coefficients. GMC macroblocks are included in predicted macroblocks. Reconstructed motion vectors of non GMC macroblocks used for predicting both luminance and chrominance have all possible combinations of half-sample and full-sample values, both for the horizontal and the vertical coordinates. Affine warping (no_of_sprite_warping_points==3) with 1/16 pixel accuracy (sprite_warping_accuracy==3) is used for reconstructing both luminance and chrominance samples in GMC macroblocks. All those combinations are used for each prediction mode in both progressive and interlaced coded VOPs.

Functional stage: MCP

Purpose: Check that decoder implements motion compensation stages with full accuracy, and affine warping stage with 1/16 pixel accuracy. And check for implementation motion vector decoding stages as defined in subclause 7.8.7.3 of ISO/IEC 14496-2/Amd-1). Except for reconstruction of Intra DC blocks, the test does not involve other decoder functions such as IDCT, inverse quantization and mismatch control. When a static decoder test is performed using the static test technique described in this document, the decoder under test shall reconstruct samples identical to those reconstructed by a reference decoder for all predicted macroblocks.

5.6.1.1.9 Test bitstream #A1GE-9

Specification: Bursty case for number of bits per macroblock with different burst location within VOP (top, bottom), followed by Bi-directional macroblocks. All motion vectors with quarter-sample components. Macroblocks outside the burst concentration have all bi-directional prediction. Number of MB/s and bitrate are the maximum allowed for the profile-and-level combination. Quarter-sample in both the horizontal and vertical directions, luminance and chrominance blocks. Maximize number of prediction blocks required to reconstruct a macroblock.

Functional stage: VLD and prediction bandwidth

Purpose: Check that decoder does not rely upon statistically small number of coded bits over local areas.

5.6.1.1.10 Test bitstream #A1GE-10

Specification: A series of consecutive progressively coded P-VOPs. As many quarter-sample components as possible in both the horizontal and vertical directions, luminance and chrominance blocks. Number of MB/s and bitrate are the maximum allowed for the profile-and-level combination. Maximize number of prediction blocks required to reconstruct a macroblock.

Functional stage: prediction bandwidth

Purpose: Check that decoder handles largest prediction bandwidth with progressively coded P-VOPs including quarter-sample interpolation. This test is somehow similar to Test bitstream GEQ#3, except that it uses progressive VOPs.

5.6.1.1.11 Test bitstream #A1GE-11

Specification: A series of consecutive progressively coded S(GMC)-VOPs. As many affine warping transformation (no_of_sprite_warping_points==3) with 1/16 pixel accuracy (sprite_warping_accuracy==3) as possible, luminance and chrominance blocks. Number of MB/s and bitrate are the maximum allowed for the profile-and-level combination. Maximize number of prediction blocks required to reconstruct a macroblock.

Functional stage: prediction bandwidth

Purpose: Check that decoder handles largest prediction bandwidth with progressively coded S(GMC)-VOPs. This test is somehow similar to Test bitstream GEG#3, except that it uses progressive VOPs.

5.6.1.1.12 Test bitstream #A1GE-12

Specification: A bitstream with a series of consecutive progressively coded B-VOPs with bi-directional macroblock motion compensation. Sequence contains many consecutive B-VOPs. Number of MB/s and bitrate are the maximum allowed for the profile-and-level combination. Use quarter-sample prediction in both the horizontal and vertical directions, for all luminance and chrominance blocks. Maximize number of prediction blocks required to reconstruct a macroblock.

Functional stage: prediction bandwidth

Purpose: Check that decoder can cope with this case of worst case bandwidth including quarter-sample interpolation. This test is somehow similar to Test bitstream GEQ#1, except that it uses progressive VOPs.

5.6.1.1.13 Test bitstream #A1GE-13

Specification: A bitstream with I-, P- and B-VOPs, with quarter-sample motion vectors that are as large as permitted by the profile-and-level combination.

Functional stage: reconstruction of motion vectors, MCP, control

Purpose: Check that decoder implements motion compensation especially for quarter-sample interpolation properly when motion vectors are very large.

5.6.1.1.14 Test bitstream #A1GE-14

Specification: A bitstream with I-, S(GMC)- and B-VOPs, with sprite trajectories and motion vectors including delta vectors for direct mode that are as large as permitted by the profile-and-level combination.

Functional stage: reconstruction of motion vectors, MCP, control

Purpose: Check that decoder implements motion compensation including global motion compensation properly when sprite trajectories and motion vectors are very large.

5.6.1.1.15 Test bitstream #A1GE-15

Specification: A series of consecutive I-VOPs with reduced_resolution_vop_enable equal to 1. The value of vop_reduced_resolution is dynamically switched between 0 and 1 for VOP by VOP.

Functional stage: Upsampling of IDCT output, block boundary filtering.

Purpose: Test the decoding process of I-VOP with reduced resolution. Proper transition between VOP with normal resolution and VOP with reduced resolution is also checked.

5.6.1.1.16 Test bitstream #A1GE-16

Specification: A series of consecutive I- and P-VOPs with reduced_resolution_vop_enable equal to 1. The value of vop_reduced_resolution is dynamically switched between 0 and 1 for VOP by VOP.

Functional stage: Upsampling of IDCT output, motion vector scaling up, motion compensation with 32x32 macroblock, block boundary filtering.

Purpose: Check the decoding process of P-VOP with reduced resolution. Proper transition between VOP with normal resolution and VOP with reduced resolution is also checked.

5.6.1.1.17 Test Bitstream #A1MHQ-1

Specification: This bitstream exercises all different horizontal quarter-pel motion vector values for vop_fcode_forward =1. The vertical motion displacements for successive macroblocks occur in the sequence {0,3,2,1,0,3,2,1,...} quarter-pels.

Functional stage: Motion vector decoding; motion compensation.

Purpose: To check that for vop_fcode_forward=1 the decoder properly handles the full range of Px, MVDx, and MVx for 16x16 block size quarter-pel motion compensated rectangular P-VOPs.

5.6.1.1.18 Test Bitstream #A1MHQ-2

Specification: This bitstream exercises all different horizontal quarter-pel motion vector values for vop_fcode_forward =2. The vertical motion displacements for successive macroblocks occur in the sequence {0,3,2,1,0,3,2,1,...} quarter-pels.

Functional stage: Motion vector decoding; motion compensation.

Purpose: To check that for vop_fcode_forward=2 the decoder properly handles the full range of Px, MVDx, and MVx for 16x16 block size quarter-pel motion compensated rectangular P-VOPs.

5.6.1.1.19 Test Bitstream #A1MHQ-3

Specification: This bitstream exercises all different horizontal quarter-pel motion vector values for vop_fcode_forward =3. The vertical motion displacements for successive macroblocks occur in the sequence {0,3,2,1,0,3,2,1,...} quarter-pels.

Functional stage: Motion vector decoding; motion compensation.

Purpose: To check that for vop_fcode_forward=3 the decoder properly handles the full range of Px, MVDx, and MVx for 16x16 block size quarter-pel motion compensated rectangular P-VOPs.

5.6.1.1.20 Test Bitstream #A1MHQ-4

Specification: This bitstream exercises all different horizontal quarter-pel motion vector values for vop_fcode_forward =4. The vertical motion displacements for successive macroblocks occur in the sequence {0,3,2,1,0,3,2,1,...} quarter-pels.

Functional stage: Motion vector decoding; motion compensation.

Purpose: To check that for vop_fcode_forward=4 the decoder properly handles the full range of Px, MVDx, and MVx for 16x16 block size quarter-pel motion compensated rectangular P-VOPs.

5.6.1.1.21 Test Bitstream #A1MHQ-5

Specification: This bitstream exercises all different horizontal quarter-pel motion vector values for vop_fcode_forward =5. The vertical motion displacements for successive macroblocks occur in the sequence {0,3,2,1,0,3,2,1,...} quarter-pels.

Functional stage: Motion vector decoding; motion compensation.

Purpose: To check that for vop_fcode_forward=5 the decoder properly handles the full range of Px, MVDx, and MVx for 16x16 block size quarter-pel motion compensated rectangular P-VOPs.

5.6.1.1.22 Test Bitstream #A1MHQ-6

Specification: This bitstream exercises all different horizontal quarter-pel motion vector values for vop_fcode_forward =6. The vertical motion displacements for successive macroblocks occur in the sequence {0,3,2,1,0,3,2,1,...} quarter-pels.

Functional stage: Motion vector decoding; motion compensation.

Purpose: To check that for vop_fcode_forward=6 the decoder properly handles the full range of Px, MVDx, and MVx for 16x16 block size quarter-pel motion compensated rectangular P-VOPs.

5.6.1.1.23 Test Bitstream #A1MHQ-7

Specification: This bitstream exercises all different horizontal quarter-pel motion vector values for vop_fcode_forward =7. The vertical motion displacements for successive macroblocks occur in the sequence {0,3,2,1,0,3,2,1,...} quarter-pels.

Functional stage: Motion vector decoding; motion compensation.

Purpose: To check that for vop_fcode_forward=7 the decoder properly handles the full range of Px, MVDx, and MVx for 16x16 block size quarter-pel motion compensated rectangular P-VOPs.

5.6.1.1.24 Test Bitstream #A1MVQ-1

Specification: This bitstream exercises all different vertical quarter-pel motion vector values for vop_fcode_forward =1. The horizontal motion displacements for successive macroblocks occur in the sequence {0,3,2,1,0,3,2,1,...} quarter-pels.

Functional stage: Motion vector decoding; motion compensation.

Purpose: To check that for vop_fcode_forward=1 the decoder properly handles the full range of Py, MVDy, and MVy for 16x16 block size quarter-pel motion compensated rectangular P-VOPs.

5.6.1.1.25 Test Bitstream #A1MVQ-2

Specification: This bitstream exercises all different vertical quarter-pel motion vector values for vop_fcode_forward =2. The horizontal motion displacements for successive macroblocks occur in the sequence {0,3,2,1,0,3,2,1,...} quarter-pels.

Functional stage: Motion vector decoding; motion compensation.

Purpose: To check that for vop_fcode_forward=2 the decoder properly handles the full range of Py, MVDy, and MVy for 16x16 block size quarter-pel motion compensated rectangular P-VOPs.

5.6.1.1.26 Test Bitstream #A1MVQ-3

Specification: This bitstream exercises all different vertical quarter-pel motion vector values for vop_fcode_forward =3. The horizontal motion displacements for successive macroblocks occur in the sequence {0,3,2,1,0,3,2,1,...} quarter-pels.

Functional stage: Motion vector decoding; motion compensation.

Purpose: To check that for vop_fcode_forward=3 the decoder properly handles the full range of Py, MVDy, and MVy for 16x16 block size quarter-pel motion compensated rectangular P-VOPs.

5.6.1.1.27 Test Bitstream #A1MVQ-4

Specification: This bitstream exercises all different vertical quarter-pel motion vector values for vop_fcode_forward =4. The horizontal motion displacements for successive macroblocks occur in the sequence {0,3,2,1,0,3,2,1,...} quarter-pels {0,3,2,1,0,3,2,1,…}.

Functional stage: Motion vector decoding; motion compensation.

Purpose: To check that for vop_fcode_forward=4 the decoder properly handles the full range of Py, MVDy, and MVy for 16x16 block size quarter-pel motion compensated rectangular P-VOPs.

5.6.1.1.28 Test Bitstream #A1MVQ-5

Specification: This bitstream exercises all different vertical quarter-pel motion vector values for vop_fcode_forward =5. The horizontal motion displacements for successive macroblocks occur in the sequence {0,3,2,1,0,3,2,1,...} quarter-pels {0,3,2,1,0,3,2,1,…}.

Functional stage: Motion vector decoding; motion compensation.

Purpose: To check that for vop_fcode_forward=5 the decoder properly handles the full range of Py, MVDy, and MVy for 16x16 block size quarter-pel motion compensated rectangular P-VOPs.

5.6.1.1.29 Test Bitstream #A1MVQ-6

Specification: This bitstream exercises all different vertical quarter-pel motion vector values for vop_fcode_forward =6. The horizontal motion displacements for successive macroblocks occur in the sequence {0,3,2,1,0,3,2,1,...} quarter-pels.

Functional stage: Motion vector decoding; motion compensation.

Purpose: To check that for vop_fcode_forward=6 the decoder properly handles the full range of Py, MVDy, and MVy for 16x16 block size quarter-pel motion compensated rectangular P-VOPs.

5.6.1.1.30 Test Bitstream #A1MVQ-7

Specification: This bitstream exercises all different vertical quarter-pel motion vector values for vop_fcode_forward =7. The horizontal motion displacements for successive macroblocks occur in the sequence {0,3,2,1,0,3,2,1,...} quarter-pels.

Functional stage: Motion vector decoding; motion compensation.

Purpose: To check that for vop_fcode_forward=7 the decoder properly handles the full range of Py, MVDy, and MVy for 16x16 block size quarter-pel motion compensated rectangular P-VOPs.

5.6.1.2 Test Bitstreams - Shape coding

5.6.1.2.1 Class 2:

5.6.1.2.1.1 Test Bitstream #A1SH-0

Specification: A series of consecutive I- and P-VOPs with binary shape only coding. The bitstream production is controlled by a random decision maker. VCV boundary MB decoder rate and bitrate are the maximum allowed for the profile @ level.

Functional stage: MV for shape, BAB type coding, MB bandwidth, reference memory bandwidth

Purpose: Check the general case of testing binary shape coding with proper test sequence for a given profile @ level structure.

5.6.1.2.2 Class 3:

5.6.1.2.2.1 Test Bitstream #A1SH-1

Specification: A series of consecutive I- and P-VOPs with binary shape and texture. The bitstream generation is controlled by a random decision maker. VCV boundary MB decoder rate and bitrate are the maximum allowed for the profile-and-level combination.

Functional stage: prediction of shape MV from texture MV

Purpose: check the general case of shape and texture coding. In particularly, tests padding and prediction of shape motion vectors from texture motion vectors with proper test sequence for a given profile @ level structure.

5.6.1.2.2.2 Test Bitstream #A1SH-2

Specification: A series of consecutive I-, P- and B-VOPs with grey scale shape and texture. The bitstream generation is controlled by a random decision maker. VCV boundary MB decoder rate and bitrate are the maximum allowed for the profile-and-level combination.

Functional stage: prediction of grey scale shape MV from texture MV

Purpose: check the general case of grey scale shape and texture coding. In particularly, tests padding and prediction of shape motion vectors from texture motion vectors with proper test sequence for a given profile @ level structure.

5.6.1.2.2.3 Test Bitstream #A1SH-3

Specification: A series of consecutive I-, P- and B-VOPs with interlaced grey scale shape and texture. The bitstream generation is controlled by a random decision maker. VCV boundary MB decoder rate and bitrate are the maximum allowed for the profile-and-level combination.

Functional stage: prediction of interlaced grey scale shape MV from texture MV

Purpose: check the general case of interlaced grey scale shape and texture coding. In particularly, tests padding and prediction of shape motion vectors from texture motion vectors with proper test sequence for a given profile @ level structure.

5.6.1.2.2.4 Test Bitstream #A1SH-4

Specification: A series of consecutive I-, P- and B-VOPs with interlaced grey scale shape and texture coding using shape adaptive DCT including all possible IDCT mismatches. The bitstream generation is controlled by a random decision maker. VCV boundary MB decoder rate and bitrate are the maximum allowed for the profile-and-level combination.

Functional stage: shape adaptive DCT including IDCT accuracy

Purpose: check the general case of interlaced grey scale shape and texture coding using shape adaptive DCT. In particularly, tests padding and prediction of shape motion vectors from texture motion vectors with proper test sequence for a given profile @ level structure. In addition test that decoders has implemented mismatch control.

5.6.1.2.2.5 Test Bitstream #A1SH-5

Specification: A series of consecutive I- and S(GMC)-VOPs with grey scale shape and texture. The bitstream generation is controlled by a random decision maker. VCV boundary MB decoder rate and bitrate are the

maximum allowed for the profile-and-level combination. Translational warping (no_of_sprite_warping_points==1) with 1/2 pixel accuracy (sprite_warping_accuracy==0) is used for constructing luminance, chrominance and grey scale shape samples in GMC macroblocks.

Functional stage: warping, pixel value interpolation for grey scale shape, and grey scale shape texture and prediction of grey scale shape MV from texture MV

Purpose: check the general case of grey scale shape and texture coding. In particularly, tests warping, padding and prediction of shape motion vectors from texture motion vectors with proper test sequence for a given profile @ level structure.

5.6.1.2.2.6 Test Bitstream #A1SH-6

Specification: A series of consecutive I- and S(GMC)-VOPs with grey scale shape and texture. The bitstream generation is controlled by a random decision maker. VCV boundary MB decoder rate and bitrate are the maximum allowed for the profile-and-level combination. Affine warping (no_of_sprite_warping_points==3) with 1/4 pixel accuracy (sprite_warping_accuracy==1) is used for constructing luminance, chrominance and grey scale shape samples in GMC macroblocks

Functional stage: warping, pixel value interpolation for grey scale shape, and grey scale shape texture and prediction of grey scale shape MV from texture MV

Purpose: check the general case of grey scale shape and texture coding. In particularly, tests warping, padding and prediction of shape motion vectors from texture motion vectors with proper test sequence for a given profile @ level structure.

5.6.1.2.2.7 Test Bitstream #A1SH-7

Specification: A series of consecutive I- and S(GMC)-VOPs with interlaced grey scale shape and texture. The bitstream generation is controlled by a random decision maker. VCV boundary MB decoder rate and bitrate are the maximum allowed for the profile-and-level combination. Isotropic warping (no_of_sprite_warping_points==2) with 1/8 pixel accuracy (sprite_warping_accuracy==2) is used for constructing luminance, chrominance and grey scale shape samples in GMC macroblocks.

Functional stage: warping, pixel value interpolation for grey scale shape, and grey scale shape texture and prediction of grey scale shape MV from texture MV

Purpose: check the general case of grey scale shape and texture coding. In particularly, tests warping, padding and prediction of shape motion vectors from texture motion vectors with proper test sequence for a given profile @ level structure.

5.6.1.2.2.8 Test Bitstream #A1SH-8

Specification: A series of consecutive I- and S(GMC)-VOPs with interlaced grey scale shape and texture. The bitstream generation is controlled by a random decision maker. VCV boundary MB decoder rate and bitrate are the maximum allowed for the profile-and-level combination. Affine warping (no_of_sprite_warping_points==3) with 1/16 pixel accuracy (sprite_warping_accuracy==3) is used for constructing luminance, chrominance and grey scale shape samples in GMC macroblocks.

Functional stage: warping, pixel value interpolation for grey scale shape, and grey scale shape texture and prediction of grey scale shape MV from texture MV

Purpose: check the general case of grey scale shape and texture coding. In particularly, tests warping, padding and prediction of shape motion vectors from texture motion vectors with proper test sequence for a given profile @ level structure.

5.6.1.3 Test Bitstreams - Error resilience

5.6.1.3.1 Test bitstream #A1ER-1

Specification: In NEWPRED mode, a reference VOP is dynamically switched a reference VOP memory according to a vop_id_for_prediction in decoding every Video Packet when newpred_segment_type is VideoPacket. The resynchronisation marker is used in this bitstream.

Functional stage: switching a reference VOP

Purpose: To test switching reference VOPs process in NEWPRED mode when newpred_segment_type is VideoPacket. In this case the decoder should get the information about the first MB number of each Video

Packet after it decodes a I-VOP. In P-VOPs, the decoder shall switch a reference and pad the image around Video Packet in decoding every Video Packet.

To decode this bitstream, the decoder has to equip the additional reference VOP memory which can store four previously decoded VOPs. In real communication system, this additional memory may be an extra item, because the decoder may not store erroneous image into the reference VOP memory. It is informative procedure to store and not to store erroneous image. The amount of additional VOP memory, which is "four" in this bitstream, depends on several non-normative items; such as the network error conditions, the strategy of the reference VOPs selection in the encoder, or the method of the memory control in the decoder.

5.6.1.3.2 Test bitstream #A1ER-2

Specification: In NEWPRED mode, a reference VOP is dynamically switched a reference VOP memory according to a vop_id_for_prediction in decoding every VOP when newpred_segment_type is VOP. The resynchronisation marker is used in this bitstream.

Functional stage: switching a reference VOP

Purpose: To test switching reference VOPs process in NEWPRED mode when newpred_segment_type is VOP. In P-VOPs, the decoder shall switch a reference in decoding every VOP.

To decode this bitstream, the decoder has to equip the additional reference VOP memory which can store four previously decoded VOPs. In real communication system, this additional memory may be an extra item, because the decoder may not store erroneous image into the reference VOP memory. It is informative procedure to store and not to store erroneous image. The amount of additional VOP memory, which is "four" in this bitstream, depends on several non-normative items; such as the network error conditions, the strategy of the reference VOPs selection in the encoder, or the method of the memory control in the decoder.

5.6.1.4 Test Bitstreams - Object based Spatial Scalability (OBSS)

5.6.1.4.1 General Test Bitstreams for OBSS

5.6.1.4.1.1 Test Bistream #A1OS-1

Specification: The bitstream has I- and P-VOP in base layer and only P-VOP in enhancement layer. The base layer is compliant bitstream of Core profile. Enhancement layer is coded with Scalability='1', hierarchy_type='0' and video_object_layer_shape='01'. The enhancement layer bitstream contains VOP coded with ref_select_code = `11`(P-VOP).

The upsampling factors for Shape/Texture are set as follows.

horizontal_sampling_factor_n : 4

horizontal_sampling_factor_m: 1

vertical_sampling_factor_n: 4

vertical_sampling_factor_m: 1

Functional Stage: Prediction process for Shape/Texture from base layer.

Purpose: This bitstream tests prediction process of shape and texture coding from base layer, i.e. Temporally coincident VOP in the reference layer (no motion vectors)

5.6.1.4.1.2 Test Bistream #A1OS-2

Specification: The bitstream has I- and P-VOP in base layer and P and B-VOP in enhancement layer. The base layer is compliant bitstream of Core profile. Enhancement layer is coded with Scalability='1', hierarchy_type='0' and video_object_layer_shape='01'. The enhancement layer bitstream contains VOP coded with ref_select_code = `00`(B-VOP).

The upsampling factors for Shape/Texture are set as follows.

horizontal_sampling_factor_n : 2

horizontal_sampling_factor_m: 1

vertical_sampling_factor_n: 2

vertical_sampling_factor_m: 1

Functional Stage: Prediction process for Shape/Texture from the enhancement layer

Purpose: This bitstream tests prediction process of Shape and Texture coding from enhancement layer. i.e. Most recently decoded enhancement VOP of the same layer.

5.6.1.4.1.3 Test bitstream #A1OS-3

Specification: The bitstream has I-, P- and B-VOP in base layer and only P-VOP in enhancement layer. The base layer is compliant bitstream of Core profile. Enhancement layer is coded with Scalability='1', hierarchy_type='0' and video_object_layer_shape='01'. The enhancement layer bitstream contains VOP coded with ref_select_code = `11`(P-VOP).

The upsampling factors for Shape/Texture are set as follows.

horizontal_sampling_factor_n : 2

horizontal_sampling_factor_m: 1

vertical_sampling_factor_n: 2

vertical_sampling_factor_m: 1

Functional Stage: Prediction process for Shape/Texture from the B-VOP coded base layer.

Purpose: This bitstream tests prediction process of Shape and Texture coding from the B-VOP coded base layer. I.e. temporally coincident VOP in the reference layer (no motion vectors)

5.6.1.4.1.4 Test bitstream #A1OS-4

Specification: The bitstream has I-, P- and B-VOP in base layer and P- and B-VOP in enhancement layer. The base layer is compliant bitstream of Core profile. Enhancement layer is coded with Scalability='1', hierarchy_type='0' and video_object_layer_shape='01'. The enhancement layer bitstream contains VOP coded with ref_select_code = `00`(B-VOP).

The upsampling factors for Shape/Texture are set as follows.

horizontal_sampling_factor_n : 8

horizontal_sampling_factor_m: 3

vertical_sampling_factor_n: 3

vertical_sampling_factor_m: 1

Functional Stage: Prediction process for Shape/Texture from the enhancement layer with B-VOP base layer coding

Purpose: This bitstream tests prediction process of Shape and Texture coding from enhancement layer, i.e. Most recently decoded enhancement VOP of the same layer, with B-VOP base layer coding.

5.6.1.4.2 Functional Test Bitstreams for OBSS

5.6.1.4.2.1 Test bitstream #A1OS-5

Specification: The bitstream has I-, P- and B-VOP in base layer and P- and B-VOP in enhancement layer. The base layer is compliant bitstream of Core profile. Enhancement layer is coded with Scalability='1', and video_object_layer_shape='10'(Binary Shape Only). Base layer also coded with 'Binary Shape Only' mode.

The upsampling factors for Shape/Texture are set as follows.

horizontal_sampling_factor_n : 1

horizontal_sampling_factor_m: 1

vertical_sampling_factor_n: 6

vertical_sampling_factor_m: 5

Functional Stage: 'Binary Shape Only' coding mode in base and enhancement layer

Purpose: This bitstream tests the 'Binary Shape Only' coding mode of enhancement layer.

5.6.1.4.2.2 Test bitstream #A1OS-6

Specification: The bitstream has I-, P- and B-VOP in base layer and only P-VOP in enhancement layer. The base layer is compliant bitstream of Core profile. Enhancement layer is coded with Object base Spatial

Scalability(Scalability='1', hierarchy_type='0' and video_object_layer_shape='01') and Shape/Texture Partial Region enhancing(enhancement_type ='1' and use_ref_shape='0'). Background_composition is set to '0' for enhancement layer coding.

Functional Stage: Shape/Texture Partial Region enhancement layer scalable coding with ROI(Region Of Interest).

Purpose: This bitstream tests Shape/Texture Partial Region enhancement layer scalable coding without Spatial Background composition.

5.6.1.4.2.3 Test Bistream #A1OS-7

Specification: The bitstream has I-, P- and B-VOP in base layer and P- and B-VOP in enhancement layer. The base layer is compliant bitstream of Core profile. Enhancement layer is coded with Object base Spatial Scalability(Scalability='1', hierarchy_type='0' and video_object_layer_shape='01') and Shape/Texture Partial Region enhancing(enhancement_type ='1' and use_ref_shape='0'). Background_composition is set to '1' for enhancement layer coding.

Functional Stage: Shape/Texture Partial Region enhancement layer scalable coding with ROI(Region Of Interest) and Spatial background composition.

Purpose: This bitstream tests Shape/Texture Partial Region enhancement layer scalable coding with Spatial Background composition.

5.6.1.4.2.4 Test bitstream #A1OS-8

Specification: The bitstream has I, P and B-VOP in base layer and P and B-VOP in enhancement layer, where Base Layer is coded as rectangular shape (video_object_layer_shape ='00') and Enhancement layer is coded as arbitrary shape (video_object_layer_shape = '01'). Texture Information in the enhancement layer is coded with texture information in the base layer and shape information is coded independently between base layer and enhancement layer.

Background_composition is applied for output enhancement layer images. The following parameters are applied for this bitstreams:

Base layer:

video_object_layer_shape='00'

Enhancement layer:

scalability='1'

hierarchy_type='0'

video_object_layer_shape='01'

enhancement_type ='1'

use_ref_shape='1'

background_composition ='1'

Functional Stage: Texture Partial Region enhancement layer scalable coding and simulcast shape coding with ROI(Region Of Interest) and spatial background composition.

Purpose: This bitstream tests Texture Partial Region enhancement layer scalable coding and simulcast shape coding in the enhancement layer with Spatial Background composition.

5.6.1.4.2.5 Test bitstream #A1OS-9

Specification: The bitstream has I-, P- and B-VOP in base layer and P- and B-VOP in enhancement layer, where both base and enhancement layer is coded with arbitrary shape. Texture Information in the enhancement layer is coded with scalable coding method texture information in the base layer and shape information is coded independently between base layer and enhancement layer.

Background_composition is applied for output enhancement layer images. The following parameters are applied for this bitstreams.

Base layer:

video_object_layer_shape='01'

Enhancement layer:

scalability='1'

hierarchy_type='0'

video_object_layer_shape='01'

enhancement_type ='1'

use_ref_shape='1'

background_composition ='0'

Functional Stage: Texture Partial Region enhancement layer scalable coding and simulcast shape coding

Purpose: This bitstream tests Texture Partial Region enhancement layer scalable coding and simulcast shape coding in the enhancement layer.

5.6.1.4.3 Performance Test Bitstreams for OBSS

5.6.1.4.3.1 Test Bistream #A1OS-10

Specification: The base layer is compliant bitstream of Core profile. The ref_select_code = '00' in B-VOP and ref_select_code = "11" in P-VOP are used for enhancement layer. Base and Enhancement layers contain binary shape and enhancement layer uses enhancement_type ='0' and use_ref_shape='0'. The max number of bitrate and Macroblock per second satisfy those of CSP@L1

Function stage: Performance of enhancement layer decoder

Purpose: The purpose of this bitstream is to verify a performance of enhancement layer decoder. The bitstream put stress for enhancement layer decoder in CSP@L1

5.6.1.4.3.2 Test bitstream #A1OS-11

Specification: The base layer is compliant bitstream of Core profile. The ref_select_code = '00' in B-VOP and ref_select_code = "11" in P-VOP are used for enhancement layer. Base and Enhancement layers contain binary shape and enhancement layer uses enhancement_type ='0' and use_ref_shape='0'. The max number of bitrate and Macroblock per second satisfy those of CSP@L2

Function stage: Performance of enhancement layer decoder

Purpose: The purpose of this bitstream is to verify a performance of enhancement layer decoder. The bitstream put stress for enhancement layer decoder in CSP@L2

5.6.1.4.4 Test bitstream #A1OS-12

Specification: The base layer is compliant bitstream of Core profile. The ref_select_code = '00' in B-VOP and ref_select_code = "11" in P-VOP are used for enhancement layer. Base and Enhancement layers contain binary shape and enhancement layer uses enhancement_type ='0' and use_ref_shape='0'. The max number of bitrate and Macroblock per second satisfy those of CSP@L3

Function stage: Performance of enhancement layer decoder

Purpose: The purpose of this bitstream is to verify a performance of enhancement layer decoder. The bitstream put stress for enhancement layer decoder in CSP@L3

5.6.1.5 Test Bitstreams - Scalable shape for scaleable textures

5.6.1.5.1 Test bitstream #A1ST-1

Specification: The bitstreams are generated using scalable shape coding for still texture (texture_object_layer_shape=="01") with change_conv_ratio_disable=="0". The bitstream is designed to use all type of BAB (SI_bab_type=="transitional BAB" or "exceptional BAB") and regenerate the lossless shape data in the base layer (alphaTh should be set to 0). In case of transitional BAB coding the transposition of the BAB is included. In addition to the above constraints the bitstream also includes the compressed data of the texture coefficients inside the shape objects (e.g. the constraints for the bitstream of the texture coefficients are as follows: quantization_type=="01" (Single_quant mode) and scan_direction=="0" (tree-depth fashion)).

Functional stage:

1. Intra CAE and upsampling (with regard to conv_ratio== 1, 2, or 4) for the base layer shape coding, transposition (with regard to scan_type) for base layer shape coding, the transitional BAB coding (the transposition included) and the exceptional BAB coding for the enhancement layer shape coding.
2. Scalable texture decoding with scalable shape

Purpose: To test a decoder for conformance with the regard of advanced core profile, the above bitstream is used to check the 1024 contexts of intra-CAE, upsampling, and transposition of BAB in the base layer for the scalable shape coding for still texture. The bitstream is also used to check whether the scalable shape coder works well with the scalable texture coder.

5.6.1.5.2 Test bitstream #A1ST-2

Specification: The bitstreams are generated using scalable shape coding for still texture (texture_object_layer_shape=="01") with change_conv_ratio_disable=="1". The bitstream is designed to use all type of BAB (SI_bab_type == "transitional BAB" or "exceptional BAB"). In case of transitional BAB coding the transposition of the BAB is included. In addition to the above constraints the bitstream also includes the compressed data of the texture coefficients inside the shape objects (e.g. the constraints for the bitstream of the texture coefficients are as follows: quantization_type=="01" (Single_quant mode), scan_direction=="1" (band-by-band fashion), start_code_enable=='0' (disabled)).

Functional stage:

1. The transitional BAB coding (the transposition included) and the exceptional BAB coding for the enhancement layer shape coding.
2. Scalable texture decoding with scalable shape

Purpose: To test a decoder for conformance with the regard of advanced core profile, the above bitstream is used to check the transposition of transitional BAB and verify the 128 contexts for transitional BAB in the vertical and horizontal scanning order, respectively. The bitstream is also used to check whether the scalable shape coder works well with the scalable texture coder.

5.6.1.5.3 Test bitstream #A1ST-3

Specification: The bitstreams are generated using scalable shape coding for still texture (texture_object_layer_shape=="01") with change_conv_ratio_disable=="1". The bitstream is designed to use all type of BAB (SI_bab_type == "transitional BAB" or "exceptional BAB"). In case of transitional BAB coding the transposition of the BAB is included. In addition to the above constraints the bitstream is also designed to use the following conditions: wavelet_filter_type=='0' (integer), wavelet_download== '0', default wavelet filter (ODD symmetry filter). The following constraints are also recommended: quantization_type=="10" (multi quantizer mode), scan_direction=='0' (tree-depth fashion), start_code_enable=='0' (disabled).

Functional stage:

1. SI_bab_type for odd symmetry filter (SI_bab_type_prob [])
2. 128 contexts for CAE of vertical and horizontal scanning order in the transitional BAB coding (The probability tables of enh_intra_v_prob[] and enh_intra_h_prob[] are used)
3. 256 contexts for even symmetry for CAE of the pixel T0 (for context0) and T1 (for context1) in the exceptional BAB coding. (sto_enh_odd_prob0[] and sto_enh_odd_prob1[] are used)
4. Scalable texture decoding with scalable shape

Purpose: To test a decoder for conformance with the regard of advanced core profile, the above bitstream is used to check the BAB type decoding in enhancement layer and to verify the two 256 contexts for exceptional BAB in case of ODD symmetry wavelet filter used. The bitstream is also used to investigate whether the scalable shape coder works well with the scalable texture coder.

5.6.1.5.4 Test bitstream #A1ST-4

Specification: The bitstreams are generated using scalable shape coding for still texture (texture_object_layer_shape=="01") with change_conv_ratio_disable=="1". The bitstream is designed to use all type of BAB (SI_bab_type == "transitional BAB" or "exceptional BAB"). In case of transitional BAB coding the transposition of the BAB is included. In addition to the above constraints the bitstream is also designed to use the following conditions: wavelet_filter_type=='0' (integer), wavelet_download== '1', wavelet filter length is

EVEN (even symmetry filter). The following constraints are also recommended : quantization_type=="10" (multi quantizer mode), scan_direction=='1' (band-by-band fashion), start_code_enable=='1' (enabled). The used filters are as follows:

Low pass g [] = {64, 192, 192, 64}

High pass h [] = {5, 15, -19, -97, 26, 350, -350, -26, 97, 19, -15, -5}

Functional stage:

1. SI_bab_type for even symmetry filter (sto_SI_bab_type_prob_even[])
2. 128 contexts for CAE of vertical and horizontal scanning order in the transitional BAB coding (The probability tables of enh_intra_v_prob[] and enh_intra_h_prob[] are used)
3. 256 contexts for even symmetry for CAE of the pixel T0 (for context0) and T1 (for context1) in the exceptional BAB coding. (sto_enh_even_prob0[] and sto_enh_even_prob1[] are used)
4. Scalable texture decoding with scalable shape

Purpose: To test a decoder for conformance with the regard of advanced core profile, the above bitstream is used to check the BAB type decoding in enhancement layer and to verify the two 256 contexts for exceptional BAB in case of EVEN symmetry wavelet filter used. The bitstream is also used to investigate whether the scalable shape coder works well with the scalable texture coder.

5.6.1.5.5 Test bitstream #A1ST-5

Specification: The bitstream is generated using wavelet tiling operation and scalable shape coding for still texture (tiling_disable=='0', texture_object_layer_shape== "01"). The bitstream is designed to use every three type of tile (texture_tile_type= "01", "10", and "11").

Functional stage: The combination of wavelet tiling and scalable shape for still texture

Purpose: To test a decoder for conformance with the regard of advanced core profile, the above bitstream is used to verify the combination of tiling operation and scalable shape coding for still texture.

5.6.1.5.6 Test bitstream #A1ST-6

Specification: The bitstream is generated using error resilience tool for still texture and scalable shape coding for still texture (texture_error_resilience_disable=='0', texture_object_layer_shape== "01").

Functional stage: The combination of error resilience tool for still texture and scalable shape for still texture

Purpose: To test a decoder for conformance with the regard of advanced core profile, the above bitstream is used to verify the combination of error resilience for still texture and scalable shape coding for still texture.

5.6.1.6 Test Bitstreams - Wavelet tiling

Following parameters are used as a common condition of defined bitstreams.

Table 25 — Common conditions of #A1WT bitstreams

decomposition_levels	4
start_code_enable	1
wavelet_filter_type	0
download_filter	0

5.6.1.6.1 Test bitstream #A1WT-1

Specification: The value of texture_object_layer_width and texture_object_layer_height are twice of tile_width and tile_height. The tiling_jump_table_enable=1, the texture_object_layer_shape='00' and the error_resilience_disable=1 are set.

Function stage: Overall

Purpose: The purpose of this bitstream is to verify the Wavelet tiling. In addition, the case of tiling_jump_table_enable=1 is verified. The SQ with tree_depth scanning is used.

5.6.1.6.2 Test bitstream #A1WT-2

Specification: The value of texture_object_layer_width and texture_object_layer_height is 4/3 times of tile_width and tile_height. The tiling_jump_table_enable=0, the texture_object_layer_shape='00' and the error_resilience_disable=1 are set.

Function stage: Overall

Purpose: The purpose of this bitstream is to verify the Wavelet tiling with different size of tile. In addition, the case of tiling_jump_table_enable=0 is verified. The MQ with band-by-band scanning is used.

5.6.1.6.3 Test bitstream #A1WT -3

Specification: The texture_object_layer_width and texture_object_layer_height is a twice of tile_width and tile_height. The tiling_jump_table_enable = 1, texture_object_layer_shape='00', target_segment_length=512 and texture_error_resilience_disable=0 are set.

Function stage: wavelet tiling + error resilience for scalable texture

Purpose: The purpose of this bitstream is to verify the Wavelet tiling with error resilience for scalable texture. In addition, the case of tiling_jump_table_enable=1 is verified. The SQ with tree_depth scanning is used.

5.6.1.7 Test Bitstreams - Error resilience for scaleable textures

Following parameters are used as a common condition of defined bitstreams.

Table 26 — Common conditions of #A1ET bitstreams

Tiling_disable	1
Texture_error_resilience_disable	0
Wavelet_filter_type	0
Wavelet_download	0
Wavelet_decomposition_levels	4
Wavelet_filter_type	0
Start_code_enable	1
Texture_object_layer_shape	00
Quantization_type	01

5.6.1.7.1 Test bitstream #A1ET-1

Specification: The parameters are scan_direction=0, header_extention_code=0 and target_segment_length=256. Packets with one and more number of texture_uintis are created.

Function stage: verifies the packetization and the segment marker tools in tree-depth case. In addition, it verifies different sizes of packets.

Purpose: The purpose of this bitstream is to verify the error resilience for tree-depth mode.

5.6.1.7.2 Test bitstream #A1ET-2

Specification: The parameters are scan_direction=1, header_extention_code=0 and target_segment_length=256. Packets with one and more number of texture_uintis are created.

Function stage: verifies the packetization and the segment marker tools in subband-by-subband case. In addition, it verifies different sizes of packets.

Purpose: The purpose of this bitstream is to verify the error resilience for subband-by-subband mode.

5.6.1.7.3 Test bitstream #A1ET-3

Specification: The parameters are scan_direction=0, header_extention_code=1 and target_segment_length=256. Packets with one and more number of texture_uintis are created.

Function stage: verifies the header_extension_code in error resilience case.

Purpose: The purpose of this bitstream is to verify the header_extention mode of error resiliency.

5.6.2 Normative Test Suites for Advanced Real-Time Simple (ARTS), Core Scaleable, Advanced Coding Efficiency (ACE), Advanced Core (AC) and Advanced Scaleable Texture profiles

In order for a decoder of a particular profile-and-level to claim compliance to the standard described by this document, the decoder shall pass successfully both the static test and the dynamic test defined in this document with all the bitstreams of the normative test suite specified for testing decoders of this particular profile-and-level.

Table 27 defines the normative test suites for each profile-and-level combination. The test suite for a particular profile-and-level combination is the list of bitstreams that are marked with an 'S', 'D' or 'X' in the column corresponding to that profile-and-level combination.

When the test suite for a profile-and-level combination does not include any bitstream of this same profile-and-level, it is not possible to test adequately compliance to the standard for decoders of that profile-and-level.

Legend:

S – Bitstream is intended for functional test

D – Bitstream is intended for dynamic test

X – Bitstream is for functional and dynamic test

Table 27 –Summary of video test bitstreams

Categories	Bitstream	Donated by	Bitstream Name	ARTS (Advanced Real-Time Simple)				Core Scalable								ACE (Advanced Coding Efficiency)				AC (Advanced Core)		Advanced Scaleable Texture		
								Simple (Base)			Core (Base)		Core Scale. (Enhance.)											
				L1	L2	L3	L4	L1	L2	L3	L1	L2	L1	L2	L3	L1	L2	L3	L4	L1	L2	L1	L2	L3
General	GE-1	Bosch	vcon_ge_1_ace_l1.bits													X								
		Bosch	vcon_ge_1_ace_l2.bits														X							
		Bosch	vcon_ge_1_ace_l3.bits															X						
		Bosch	vcon_ge_1_ace_l4.bits																X					
	GE-2	Hitachi	vcon-ge2-ACEL1.bits													X								
		Hitachi	vcon-ge2-ACEL2.bits														X							
		Hitachi	vcon-ge2-ACEL3.bits															X						
		Hitachi	vcon-ge2-ACEL4.bits																X					
	GE-3	UH	ge-3_ace_l1.bits													X								
		UH	ge-3_ace_l2.bits														X							
		UH	ge-3_ace_l3.bits															X						
		UH	ge-3_ace_l4.bits																X					
	GE-4	Hitachi	vcon-ge4-ACEL1.bits													S	S	S	S					
	GE-6	NTT	GE-6-L1.cmp													S	S	S	S					
	GE-8	NTT	GE-8-L1.cmp													S	S	S	S					
	GE-10	UH	ge-10_ace.bits													S	S	S	S					
	GE-11	Hitachi	vcon-ge11-ACEL1.bits													X								
		Hitachi	vcon-ge11-ACEL2.bits														X							
		Hitachi	vcon-ge11-ACEL3.bits															X						
		Hitachi	vcon-ge11-ACEL4.bits																X					
	GE-12	UH	ge-12_ace_l1.bits													X								
		UH	ge-12_ace_l2.bits														X							
		UH	ge-12_ace_l3.bits															X						
		UH	ge-12_ace_l4.bits																X					
	GE-13	Bosch	vcon_ge_13_ace_l1.bits													X								

Categories	Bitstream	Donated by	Bitstream Name	ARTS (Advanced Real-Time Simple)				Core Scalable								ACE (Advanced Coding Efficiency)				AC (Advanced Core)		Advanced Scaleable Texture		
								Simple (Base)			Core (Base)		Core Scale. (Enhance.)											
				L1	L2	L3	L4	L1	L2	L3	L1	L2	L1	L2	L3	L1	L2	L3	L4	L1	L2	L1	L2	L3
		Bosch	vcon_ge_13_ace_l2.bits														X							
		Bosch	vcon_ge_13_ace_l3.bits															X						
		Bosch	vcon_ge_13_ace_l4.bits																X					
	GE-13	Mitsubishi	vcon-ge13-L1.bits	S																				
		Mitsubishi	vcon-ge13-L2.bits		S																			
		Mitsubishi	vcon-ge13-L3.bits			S																		
	GE-14	UH	ge-14_ace_l1.bits													X								
		UH	ge-14_ace_l2.bits														X							
		UH	ge-14_ace_l3.bits															X						
		UH	ge-14_ace_l4.bits																X					
	GE-16	Mitsubishi	vcon-ge16-L1.bits	S																				
		Mitsubishi	vcon-ge16-L2.bits		S																			
		Mitsubishi	vcon-ge16-L3.bits			S																		
	GE-16	Hitachi	vcon-ge16-ACEL1.bits													S	S	S	S					
	GE-18	Hitachi	vcon-ge18-ACEL1.bits													X								
		Hitachi	vcon-ge18-ACEL2.bits														X							
		Hitachi	vcon-ge18-ACEL3.bits															X						
		Hitachi	vcon-ge18-ACEL4.bits																X					
	GE-19	Hitachi	vcon-ge19-ACEL1.bits													S	S	S	S					
	GE-20	Hitachi	vcon-ge20-ACEL1.bits													S	S	S	S					
	GE-21	Hitachi	vcon-ge21-ACEL1.bits													S	S	S	S					
	GE-22	Hitachi	vcon-ge22-ACEL1.bits													S	S	S	S					
	GE-23	Bosch	vcon_ge_23_ace_l1.bits													X								
		Bosch	vcon_ge_23_ace_l2.bits														X							
		Bosch	vcon_ge_23_ace_l2.bits															X						
		Bosch	vcon_ge_23_ace_l2.bits																X					
	GE-24	Hitachi	vcon-ge24-ACEL1.bits													S	S	S	S					
	GE-25	Hitachi	vcon-ge25-ACEL1.bits													S	S	S	S					
	A1GE-1	Bosch	vcon_a1ge_1_ace_l1.bits													X								
		Bosch	vcon_a1ge_1_ace_l2.bits														X							
		Bosch	vcon_a1ge_1_ace_l3.bits															X						
		Bosch	vcon_a1ge_1_ace_l4.bits																X					
	A1GE-2	UH	a1ge-2_ace_l1.bits													X								
		UH	a1ge-2_ace_l2.bits														X							
		UH	a1ge-2_ace_l3.bits															X						
		UH	a1ge-2_ace_l4.bits																X					
	A1GE-3	NTT	A1GE-03-L1.cmp													X								

Categories	Bitstream	Donated by	Bitstream Name	ARTS (Advanced Real-Time Simple)				Core Scalable								ACE (Advanced Coding Efficiency)				AC (Advanced Core)		Advanced Scaleable Texture		
								Simple (Base)			Core (Base)		Core Scale. (Enhance.)											
				L1	L2	L3	L4	L1	L2	L3	L1	L2	L1	L2	L3	L1	L2	L3	L4	L1	L2	L1	L2	L3
		NTT	A1GE-03-L2.cmp														X							
		NTT	A1GE-03-L3.cmp															X						
		NTT	A1GE-03-L4.cmp																X					
	A1GE-4	UH	a1ge-4_ace.bits													S	S	S	S					
	A1GE-5	NTT	A1GE-05-L1.cmp													S	S	S	S					
	A1GE-6	NTT	A1GE-06-L2.cmp													S	S	S	S					
	A1GE-7	NTT	A1GE-07-L3.cmp													S	S	S	S					
	A1GE-8	NTT	A1GE-08-L4.cmp													S	S	S	S					
	A1GE-9	UH	a1ge-9_ace_l1.bits													X								
		UH	a1ge-9_ace_l2.bits														X							
		UH	a1ge-9_ace_l3.bits															X						
		UH	a1ge-9_ace_l4.bits																X					
	A1GE-10	Bosch	vcon_a1ge_10_ace_l1.bits													X								
		Bosch	vcon_a1ge_10_ace_l2.bits														X							
		Bosch	vcon_a1ge_10_ace_l3.bits															X						
		Bosch	vcon_a1ge_10_ace_l4.bits																X					
	A1GE-11	NTT	A1GE-11-L1.cmp													X								
		NTT	A1GE-11-L2.cmp														X							
		NTT	A1GE-11-L3.cmp															X						
		NTT	A1GE-11-L4.cmp																X					
	A1GE-12	UH	a1ge-12_ace_l1.bits													X								
		UH	a1ge-12_ace_l2.bits														X							
		UH	a1ge-12_ace_l3.bits															X						
		UH	a1ge-12_ace_l4.bits																X					
	A1GE-13	Bosch	vcon_a1ge_13_ace_l1.bits													X								
		Bosch	vcon_a1ge_13_ace_l2.bits														X							
		Bosch	vcon_a1ge_13_ace_l3.bits															X						
		Bosch	vcon_a1ge_13_ace_l4.bits																X					
	A1GE-14	NTT	A1GE-14-L1.cmp													X								
		NTT	A1GE-14-L2.cmp														X							
		NTT	A1GE-14-L3.cmp															X						
		NTT	A1GE-14-L4.cmp																X					
	A1GE-15	Fujitsu	vcon-A1GE-15-ARTS-L1.bits	X																				
		Fujitsu	vcon-A1GE-15-ARTS-L2.bits		X																			
		Fujitsu	vcon-A1GE-15-ARTS-L3.bits			X																		
		Fujitsu	vcon-A1GE-15-ARTS-L4.bits				X																	

Categories	Bitstream	Donated by	Bitstream Name	ARTS (Advanced Real-Time Simple)				Core Scalable								ACE (Advanced Coding Efficiency)				AC (Advanced Core)		Advanced Scaleable Texture		
								Simple (Base)			Core (Base)		Core Scale. (Enhance.)											
				L1	L2	L3	L4	L1	L2	L3	L1	L2	L1	L2	L3	L1	L2	L3	L4	L1	L2	L1	L2	L3
	A1GE-16	Fujitsu	vcon-A1GE-16-ARTS-L1.bits	X																				
		Fujitsu	vcon-A1GE-16-ARTS-L2.bits		X																			
		Fujitsu	vcon-A1GE-16-ARTS-L3.bits			X																		
		Fujitsu	vcon-A1GE-16-ARTS-L4.bits				X																	
Binary Shape	SH-1	Siemens	sh1_ace_enc.bits													X								
		Siemens	sh1_ace_enc.bits														X							
		Siemens	sh1_ace_enc.bits															X						
		Siemens	sh1_ace_enc.bits																X					
		Sony	Vcon-sh1.bits																	S	S			
	SH-2	Sony	Vcon-sh2.bits																	S				
	SH-3	Sony	Vcon-sh3.bits																		S			
	SH-4																							
	SH-5																							
	SH-6																							
	SH-7-1	Toshiba	vcon-sh7-1.cmp								S									S				
	SH-7-2	Toshiba	vcon-sh7-2.cmp								S									S				
	SH-8-1	Toshiba	vcon-sh8-1.cmp									S									S			
	SH-8-2	Toshiba	vcon-sh8-2.cmp									S									S			
	SH-9																							
	SH-10																							
	SH-11																							
Shape	A1SH-0	HHI	a1sh0_L1.bits													X								
		HHI	a1sh0_L2.bits														X							
		HHI	a1sh0_L3.bits															X						
		HHI	a1sh0_L4.bits																X					
	A1SH-1	Siemens	a1sh1_ace_L1_enc.bits													X								
		Siemens	a1sh1_ace_L2_enc.bits														X							
		Siemens	a1sh1_ace_L3_enc.bits															X						
		Siemens	a1sh1_ace_L4_enc.bits																X					
	A1SH-2	Siemens	a1sh2_ace_L1_enc.bits													X								
		Siemens	a1sh2_ace_L2_enc.bits														X							
		Siemens	a1sh2_ace_L3_enc.bits															X						
		Siemens	a1sh2_ace_L4_enc.bits																X					
	A1SH-3	HHI	a1sh3_L1.bits													X								
		HHI	a1sh3_L2.bits														X							
		HHI	a1sh3_L3.bits															X						
		HHI	a1sh3_L4.bits																X					
	A1SH-4	HHI	a1sh4_L1.bits													X								

Categories	Bitstream	Donated by	Bitstream Name	ARTS (Advanced Real-Time Simple)				Core Scalable								ACE (Advanced Coding Efficiency)				AC (Advanced Core)		Advanced Scaleable Texture		
								Simple (Base)			Core (Base)		Core Scale. (Enhance.)												
				L1	L2	L3	L4	L1	L2	L3	L1	L2	L1	L2	L3	L1	L2	L3	L4	L1	L2	L1	L2	L3
		HHI	a1sh4_L2.bits														X							
		HHI	a1sh4_L3.bits															X						
		HHI	a1sh4_L4.bits																X					
	A1SH-5	NTT	A1SH-5-L1.cmp													X								
	A1SH-6	NTT	A1SH-6-L2.cmp														X							
	A1SH-7	Hitachi	vcon-a1sh7-ACEL3.bits															X						
	A1SH-8	Hitachi	vcon-a1sh8-ACEL4.bits																X					
Scalability	SCC-1	Sharp	vcon-scc1.cmp																	S	S			
	SCC-1_e	Sharp	vcon-scc1_e.cmp																	S	S			
	SCC-2	Sharp	vcon-scc2.cmp																	S	S			
	SCC-2_e	Sharp	vcon-scc2_e.cmp																	S	S			
	SCC-3	Sharp	vcon-scc3.cmp																	S	S			
	SCC-3_e	Sharp	vcon-scc3_e.cmp																	S	S			
	SCC-4	Sharp	vcon-scc4.cmp																	S	S			
	SCC-4_e	Sharp	vcon-scc4_e.cmp																	S	S			
	SCC-5	Sharp	vcon-scc5.cmp																	S	S			
	SCC-5_e	Sharp	vcon-scc5_e.cmp																	S	S			
	SCC-6	Sharp	vcon-scc6.cmp																	S	S			
	SCC-6_e	Sharp	vcon-scc6_e.cmp																	S	S			
	SCC-7	Sharp	vcon-scc7.cmp																	D				
	SCC-7_e	Sharp	vcon-scc7_e.cmp																	D				
	SCC-8	Sharp	vcon-scc8.cmp																		D			
	SCC-8_e	Sharp	vcon-scc8_e.cmp																		D			
Error resilience	er-1	NTT	ER-01-L1.cmp													S	S	S	S					
	er-1	Toshiba	Vcon-er1.cmp			S																		
	er-2-1	Toshiba	Vcon-er2-1.cmp	S																				
	er-2-2	Toshiba	Vcon-er2-2.cmp		S																			
	er-2-3	Toshiba	Vcon-er2-3.cmp			S																		
	er-3-1	Toshiba	Vcon-er3-1.cmp	S																				
	er-3-2	Toshiba	Vcon-er3-2.cmp		S																			
	er-3-3	Toshiba	Vcon-er3-3.cmp			S																		
	A1ER-1	Oki	vcon-a1er1-1.bits	S																				
		Oki	vcon-a1er1-2.bits		S																			
		Oki	vcon-a1er1-3.bits			S																		
		Oki	vcon-a1er1-4.bits				S																	
	A1ER-2	NTT	vcon-a1er2-1.cmp	S																				
		NTT	vcon-a1er2-2.cmp		S																			
		NTT	vcon-a1er2-3.cmp			S																		
		NTT	vcon-a1er2-4.cmp				S																	
Object based Scalability	A1OS-1	Samsung	vcon-a1os1.bits								S	S												

Categories	Bitstream	Donated by	Bitstream Name	ARTS (Advanced Real-Time Simple)				Core Scalable								ACE (Advanced Coding Efficiency)				AC (Advanced Core)		Advanced Scaleable Texture		
								Simple (Base)			Core (Base)		Core Scale. (Enhance.)												
				L1	L2	L3	L4	L1	L2	L3	L1	L2	L1	L2	L3	L1	L2	L3	L4	L1	L2	L1	L2	L3
	A1OS-1_e	Samsung	vcon-a1os1_e.bits										S	S	S									
	A1OS-2	Samsung	vcon-a1os2.bits								S	S												
	A1OS-2_e	Samsung	vcon-a1os2_e.bits										S	S	S									
	A1OS-3	Samsung	vcon-a1os3.bits								S	S												
	A1OS-3_e	Samsung	vcon-a1os3_e.bits										S	S	S									
	A1OS-4	Samsung	vcon-a1os4.bits								S	S												
	A1OS-4_e	Samsung	vcon-a1os4_e.bits										S	S	S									
	A1OS-5	Samsung	vcon-a1os5.bits								S	S												
	A1OS-5_e	Samsung	vcon-a1os5_e.bits										S	S	S									
	A1OS-6	Samsung	vcon-a1os6.bits								S	S												
	A1OS-6_e	Samsung	vcon-a1os6_e.bits										S	S	S									
	A1OS-7	Samsung	vcon-a1os7.bits								S	S												
	A1OS-7_e	Samsung	vcon-a1os7_e.bits										S	S	S									
	A1OS-8	Sony	vcon-a1os8.bits					S	S	S														
	A1OS-8_e	Sony	vcon-a1os8_e.bits										S	S	S									
	A1OS-9	Sony	vcon-a1os9.bits								S	S												
	A1OS-9_e	Sony	vcon-a1os9_e.bits										S	S	S									
	A1OS-10	Samsung	vcon-a1os10.bits								S	S												
	A1OS-10_e	Samsung	vcon-a1os10_e.bits										S	S	S									
	A1OS-11	Samsung	vcon-a1os11.bits								S	S												
	A1OS-11_e	Samsung	vcon-a1os11_e.bits											S	S									
	A1OS-12	Samsung	vcon-a1os12.bits								S	S												
	A1OS-12_e	Samsung	vcon-a1os12_e.bits												S									
Scalable shape for scalable textures	A1ST-1	Samsung	vcon-a1st1.bits																	S	S	S	S	S
	A1ST-2	Samsung	vcon-a1st2.bits																	S	S	S	S	S
	A1ST-3	Samsung	vcon-a1st3.bits																	S	S	S	S	S
	A1ST-4	Samsung	vcon-a1st4.bits																	S	S	S	S	S
	A1ST-5	Samsung	vcon-a1st5.bits																	S	S	S	S	S
	A1ST-6	Samsung																		S	S	S	S	S
Wavelet Tiling	A1WT-1	Sharp	vcon-a1wt1.bits																	S	S	S	S	S
	A1WT-2	Sharp	vcon-a1wt2.bits																	S	S	S	S	S
	A1WT-3	Sharp																		S	S	S	S	S
Error resilience for scalable textures	A1ET-1	Sarnoff																		S	S	S	S	S
	A1ET-2	Sarnoff																		S	S	S	S	S
	A1ET-3	Sarnoff																		S	S	S	S	S

Categories	Bitstream	Donated by	Bitstream Name	ARTS (Advanced Real-Time Simple)				Core Scalable								ACE (Advanced Coding Efficiency)				AC (Advanced Core)		Advanced Scaleable Texture		
								Simple (Base)			Core (Base)		Core Scale. (Enhance.)												
				L1	L2	L3	L4	L1	L2	L3	L1	L2	L1	L2	L3	L1	L2	L3	L4	L1	L2	L1	L2	L3
General	A1MHQ-1	Sorenson	qtrpel1h.bits													X								
General	A1MHQ-2	Sorenson	qtrpel2h.bits													X								
General	A1MHQ-3	Sorenson	qtrpel3h.bits													X								
General	A1MHQ-4	Sorenson	qtrpel4h.bits													X								
General	A1MHQ-5	Sorenson	qtrpel5h.bits													X								
General	A1MHQ-6	Sorenson	qtrpel6h.bits													X								
General	A1MHQ-7	Sorenson	qtrpel7v.bits													X								
General	A1MVQ-1	Sorenson	qtrpel1v.bits													X								
General	A1MVQ-2	Sorenson	qtrpel2v.bits													X								
General	A1MVQ-3	Sorenson	qtrpel3v.bits													X								
General	A1MVQ-4	Sorenson	qtrpel4v.bits													X								
General	A1MVQ-5	Sorenson	qtrpel5v.bits													X								
General	A1MVQ-6	Sorenson	qtrpel6v.bits													X								
General	A1MVQ-7	Sorenson	qtrpel7v.bits													X								
General	A1MVQ-5	Sorenson	qtrpel5v.bits													X								
General	A1MVQ-6	Sorenson	qtrpel6v.bits													X								
General	A1MVQ-7	Sorenson	qtrpel7v.bits													X								

6 Audio

6.1 Terms and Definitions

The following audio related terms and definitions will be used throughout this clause:

conformance data – Conformance test sequences and conformance tools.

conformance tools – Tools which are provided within an electronic annex of ISO/IEC 14496-4 to check certain conformance criteria.

conformance test sequences – The superset of compressed data and its reference waveforms provided as examples within an electronic annex of this document.

compressed data – Encoded data according to ISO/IEC 14496-3.

reference waveforms – Decoded counterparts of the compressed data.

6.2 Introduction

This **clause** specifies how tests can be designed to verify whether compressed data and decoders meet requirements specified by ISO/IEC 14496-3. In this part, encoders are not addressed specifically. An encoder can be stated as an ISO/IEC 14496-3 encoder if it generates compressed data compliant with the syntactic and semantic 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 document summarizes the requirements; cross references them to characteristics, and defines how conformance with them can be tested. Guidelines are given on how to construct test suites to check or verify decoder conformance. Some examples of compressed data implemented according to these guidelines are provided as an electronic annex to this document together with their uncompressed counterparts (reference waveforms).

6.3 Audio Conformance Points

All audio decoders except the LATM-based decoders are part of the MPEG-4 framework. Table 28 gives an overview about the interfaces that have to be provided to test the audio decoders using the MPEG-4 System.

Table 28 – Conformance points

conformance point / interface	data flow direction	description / reference
AudioSpecificConfig	in	audio related decoder specific information, see ISO/IEC 14496-3:2001 (subclause 1.6.2.1 AudioSpecificConfig)
audio access units	in	audio related bitstream payload, see ISO/IEC 14496-1:2000 (subclause 7.2.3 Access Units (AU))
BIFS/AudioSource node	in	see ISO/IEC 14496-1:2000 (subclause 9.4.2.12 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 14496-1:2000 (subclause 7.2.8 Composition Units (CU))

Figure 4 gives an overview about the test bench (MPEG-4 System), the system under test (Audio decoder), and the interfaces between them. Figure 5 gives a more detailed view on the audio decoder, consisting of error protection (EP) decoder and audio core decoder.

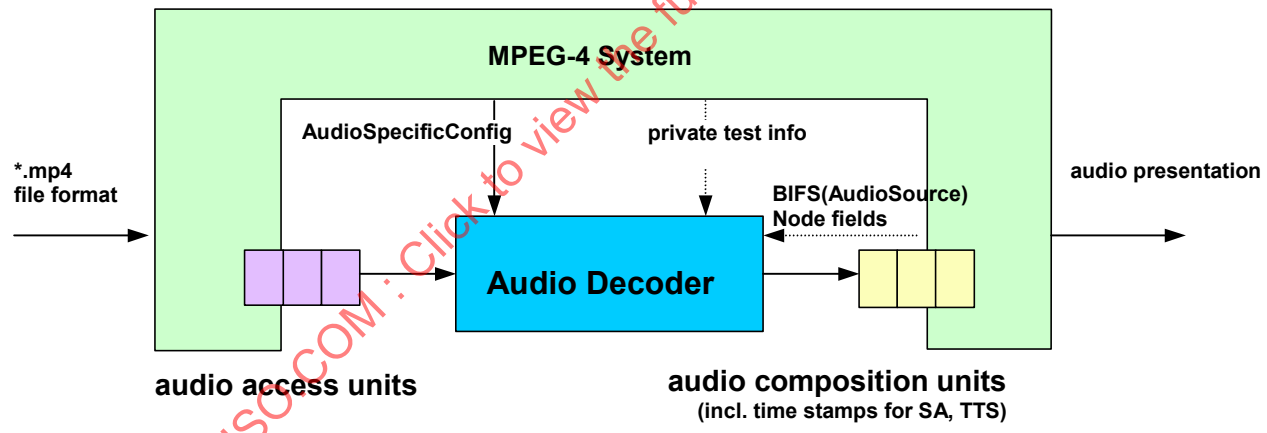


Figure 4 – Audio Conformance Points

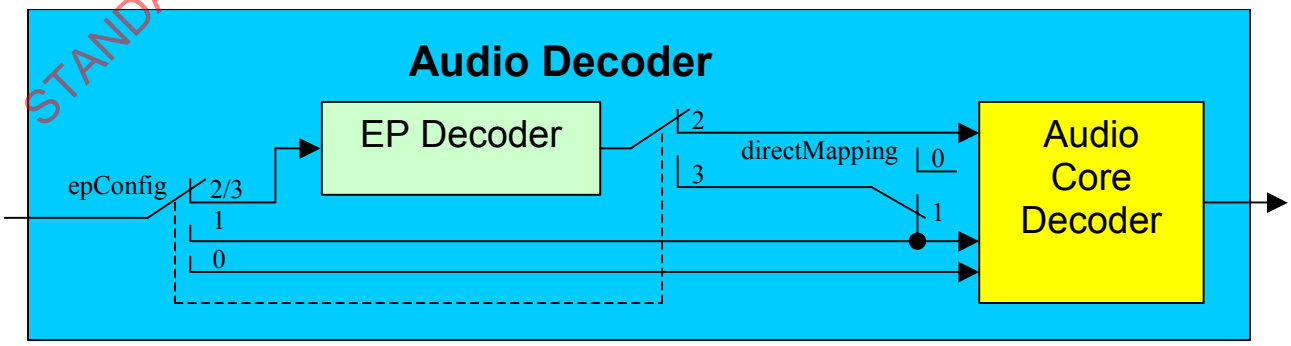


Figure 5 – Audio decoder structure

Subclause 6.6 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).

Subclause 6.7 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.

6.4 Audio 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 subclause 6.12.

6.5 Conformance data

6.5.1 File name conventions

For all conformance test sequences, the file name convention given in Table 29 is used.

Table 29 – 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>
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>
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>]

<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.

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

compressed	MPEG-4 file format	.mp4
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

6.5.2 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.

6.6 Audio Object Types

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

6.6.1 General

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

6.6.1.1 Compressed Data

6.6.1.1.1 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)

6.6.1.1.1.1 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 30 – 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 31
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 31 – 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 32 – 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	

6.6.1.1.2 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.

6.6.1.2 Decoders

6.6.1.2.1 Characteristics

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

6.6.1.2.2 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.

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.

6.6.1.2.2.1 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.

6.6.1.2.2.1.1 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.

6.6.1.2.2.2 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

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

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

6.6.1.2.2.3 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));
        }
    }
```

6.6.1.2.2.4 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 64 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.

6.6.2 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.

6.6.3 AAC-based scalable configurations

6.6.3.1 Compressed data

6.6.3.1.1 Characteristics

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

6.6.3.1.1.1 Layer configuration

number of non-AAC layers

number of AAC layers

6.6.3.1.1.2 AudioSpecificConfig

See description for individual object types.

6.6.3.1.1.3 Bitstream payload

See description for individual object types.

6.6.3.1.2 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.

6.6.3.1.2.1 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 33 are valid.

Table 33 – 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).

6.6.3.1.2.2 AudioSpecificConfig

6.6.3.1.2.2.1 AudioSpecificConfig()

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

6.6.3.1.2.3 Bitstream payload

6.6.3.1.2.3.1 tvq_scalable_main_header()

tns_data_present: Shall be 0.

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

6.6.4.1 Compressed data

6.6.4.1.1 Characteristics

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

6.6.4.1.1.1 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

6.6.4.1.1.2 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

6.6.4.1.2 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.

6.6.4.1.2.1 AudioSpecificConfig

6.6.4.1.2.1.1 AudioSpecificConfig()

audioObjectType: Shall be encoded according to the AAC object type (see Table 36).

samplingFrequencyIndex: Shall be encoded with the values specified in Table 34.

samplingFrequency: Shall be encoded with the values specified in Table 34.

Table 34 – 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	

channelConfiguration: shall be encoded with the values specified in Table 35. 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 Profiles”).

Table 35– 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

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.

6.6.4.1.2.2 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.

6.6.4.1.2.3 program_config_element()

program_configuration_element()'s in access units shall be ignored. Therefore, PCEs transmitted in Access Units cannot be used to convey decoder configuration information. The PCE in the GASpecificConfig() describes the decoder information for the elementary stream under consideration.

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: shall only be encoded with a value of 1 if a 3 front/2 rear 5-channel program 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.

6.6.4.1.2.1.4 ErrorProtectionSpecificConfig()

number_of_concatenated_frame: Shall be one.

For details see also subclause 6.7.

6.6.4.1.2.2 Bitstream payload

6.6.4.1.2.2.1 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

6.6.4.1.2.2.2 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.

6.6.4.1.2.2.3 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.

6.6.4.1.2.2.4 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	{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 conformant bitstream shall consist of only meaningful window_sequence transitions. However, decoders are required to handle non-meaningful window_sequence transitions as well. Test bitstreams al03 and as17 are provided respectively for Main and Low-Complexity profiles to test decoder performance on non-meaningful transitions (see subclause 6.6.4.1.2.2.1). 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); 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.

6.6.4.1.2.2.5 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.

6.6.4.1.2.2.6 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 subclause 6.6.4.1.2.2.17.

6.6.4.1.2.2.7 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.

6.6.4.1.2.2.8 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.

6.6.4.1.2.2.9 fill_element()

count: no restrictions apply.

esc_count: no restrictions apply.

6.6.4.1.2.2.10 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().

6.6.4.1.2.2.11 aac_scalable_main_header()

ics_reserved_bit: see subclause 6.6.4.1.2.2.4.

window_sequence: see subclause 6.6.4.1.2.2.4.

window_shape: see subclause 6.6.4.1.2.2.4.

max_sfb: see subclause 6.6.4.1.2.2.4.

scale_factor_grouping: see subclause 6.6.4.1.2.2.4.

ms_mask_present: see subclause 6.6.4.1.2.2.4.

tns_channel_mono_layer: no restrictions apply.

tns_data_present: see subclause 6.6.4.1.2.2.13.

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

6.6.4.1.2.2.12 aac_scalable_extension_header()

max_sfb: see subclause 6.6.4.1.2.2.4.

ms_mask_present: see subclause 6.6.4.1.2.2.4.

tns_data_present: see subclause 6.6.4.1.2.2.13.

6.6.4.1.2.2.13 diff_control_data()

diff_control: no restrictions apply.

6.6.4.1.2.2.14 diff_control_lr()

diff_control_lr: no restrictions apply.

6.6.4.1.2.2.15 individual_channel_stream()

global_gain: no restrictions apply.

pulse_data_present: shall be encoded with a value of 0 for AAC scalable or if window_sequence is EIGHT_SHORT_SEQUENCE.

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.

6.6.4.1.2.2.16 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.

6.6.4.1.2.2.17 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.

6.6.4.1.2.2.18 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.

6.6.4.1.2.2.19 ltp_data()

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

6.6.4.1.2.2.20 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'.

6.6.4.1.2.2.21 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.

6.6.4.1.2.2.22 dynamic_range_info()

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

6.6.4.1.2.2.23 excluded_channels()

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

6.6.4.1.2.2.24 ms_data()

ms_used: see subclause 6.6.4.1.2.2.3.

6.6.4.2 Decoders

6.6.4.2.1 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 36 — 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 14496-3:2001. Table 37 gives an overview.

Table 37 – Overview about the AAC object types

Audio Object Type	Object Type ID	13818-7 LC	PNS	LTP	TLSS	Low Delay AAC	Error Robust
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

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 38 shows these dependencies.

Table 38 – 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 39 — 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

6.6.4.2.1.1 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.

6.6.4.2.1.2 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.

6.6.4.2.1.3 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.

6.6.4.2.1.4 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.

6.6.4.2.1.5 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.

6.6.4.2.2 Test procedure

The test procedures specified in Table 42 has to be applied. The RMS test procedure always includes the LSB test. Table 40 provides the references to the according test specifications.

Table 40 – References to the test precedure descriptions

Name of the test procedure as used in Table 42	Reference to the test specification
RMS	subclause 6.6.1.2.2.1
PNS	subclause 6.6.1.2.2.4

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.

6.6.4.2.3 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 41 and Table 42. In the case that the ChannelConfiguration equals zero the program_config_element() is defined in Table 43. 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. 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 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 subclause 6.6.4.1.2.2.1 for details).

Table 41 – 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
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
al15	music	192/384	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	y	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

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
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	
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	

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
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	
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	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 42 – 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_ad100	sine sweep	64	23	*	1	0,1	0	-	0	0	0	-	-	-	y	n	n	?	n	RMS
er_ad103	test mix	64	23	*	1	0,1	0	-	0	1	0	-	-	-	?	?	n	?	fs1	none
er_ad107	test mix	64	23	*	1	0,1	0	-	0	1	1	-	-	-	?	?	n	?	fs2	none
er_ad109	noise	64	23	*	1	0,1	0	-	0	0	0	-	-	-	n	n	n	n	y	PNS-1
er_ad110	sine sweep	64	23	*	1	0,1	1	-	0	0	0	-	-	-	y	n	n	?	n	RMS
er_ad111	test mix	64	23	*	1	0,1	1	-	0	0	0	-	-	-	?	?	n	?	fs2	none
er_ad115	test mix	64	23	*	1	0,1	1	-	0	0	1	-	-	-	?	?	n	?	fs1	none
er_ad202	test mix	128	23	*	2	0,1	0	-	1	0	0	y	y	-	?	?	n	?	fs1	none
er_ad206	test mix	128	23	*	2	0,1	0	-	1	0	1	y	y	-	?	?	n	?	fs2	none
er_ad214	test mix	128	23	*	2	0,1	1	-	1	1	0	y	y	-	?	?	n	?	fs1	none
er_ad218	test mix	128	23	*	2	0,1	1	-	1	1	1	y	y	-	?	?	n	?	fs2	none
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
er_ac118	sine sweep	40 24 40 40	20 20 20 20	2 2 2 2	1 1 2 2	0,1	0 0 0 0	-	0 0 1 1	1 1 0 0	0 1 0 1	- - n n	- - y y	y - - -	y - - -	n - n -	n n n n	n - - -	n n n n	RMS
er_ac119	sine sweep	30 16 96 20	20 20 20 20	7 7 7 7	1 1 2 2	0,1	0 0 0 0	-	0 0 1 1	1 1 0 0	0 1 0 1	- - n n	- - y y	y - - -	y - - -	n - n -	n n n n	n - - -	n n n n	RMS
er_ac121	music	32 32 32 32	20 20 20 20	3 3 3 3	2 2 2 2	0	0 0 0 0	-	0 0 0 0	0 0 0 0	0 0 0 0	y y n n	y y y y	n - - -	n - - -	n - - -	n n n n	- - - -	n n n n	none
er_ac123	test mix	40 64	20 20	4 4	2 2	0,1	0 0	-	1 1	1 1	0 1	n n	y y	? -	? -	? -	n n	n -	y n	none
er_ac211	test mix	6 40 64	24 20 20	11 6 6	1 1 2	0,1	- 1 1	- 0 -	- 0 0	- 1 0	- 0 1	- - n	- - y	- ? -	- ? -	- ? ?	- n n	- - -	- y n	none
er_ac221	test mix	6.2 64 128	24 20 20	12 7 7	1 2 2	0,1	- 1 1	- 0 -	- 0 0	- 0 1	- 0 1	- n n	- y y	- ? -	- ? -	- ? -	- n n	- - -	- n n	none
er_ac311	test mix	8 40 64	21 20 20	5 5 5	1 1 2	0,1	0 0 0	-	- 1 1	- 0 1	- 0 1	- - n	- - y	? -	? -	- ? ?	- n n	n - -	- n n	none
er_ac321	test mix	12 64 96	21 20 20	6 6 6	1 2 2	0,1	1 1 1	-	- 1 1	- 1 0	- 0 1	- n n	- y y	? -	? -	- ? -	- n n	n - -	- y n	none

Table 43 – 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	-
al15	3	n y y	0	-	0	-	1	1	y
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	-

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
as13	2	n y	1	n	0	-	1	0	-
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

6.6.5 TwinVQ and ER_TwinVQ

6.6.5.1 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).

6.6.5.2 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)

6.6.5.3 Procedure to Test Bitstream Conformance

6.6.5.3.1 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.

6.6.5.3.2 Decoding of the payload

6.6.5.3.2.1 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.

6.6.5.3.2.2 parsing tvq_scalable_extension_header()

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

6.6.5.3.2.3 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.

6.6.5.4 Decoder Characteristics

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

6.6.5.5 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 44, Table 45 and Table 46. 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 subclause 6.6.1.2.2.1)

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.

6.6.5.6 Descriptions of the audio test bitstreams

Table 44 — 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 45 — 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 46 — 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 47 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

6.6.6 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. However, only mono or 2-channel stereo objects are supported.

6.6.6.1 Compressed data

6.6.6.1.1 Characteristics

6.6.6.1.1.1 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

6.6.6.1.1.2 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

6.6.6.1.2 Test procedure

6.6.6.1.2.1 AudioSpecificConfig

The following restrictions apply to AudioSpecificConfig:

AudioObjectType: Shall be encoded with the value 22

SamplingFrequencyIndex: Shall be encoded with the following values:

Table 48

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 49

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 50

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

6.6.6.1.2.2 Audio Acess Units

6.6.6.1.2.2.1 bsac_base_element ()

frame_length: must be larger than or equal to 4

6.6.6.1.2.2.2 bsac_header ()

header_length: ((header_length+8)*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.

6.6.6.1.2.2.3 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.

6.6.6.2 Decoders

6.6.6.2.1 Characteristics

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

Table 51 – 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

6.6.6.2.2 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 52. 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 52. 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 16kHz.

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 52. 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 16kHz.

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 52. The lower **highestLayer** is to be used for sampling rates of 16kHz and below, and the higher **highestLayer** is to be used at sampling rates above 16kHz.

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 the subclause 6.6.1.2.2.1. 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.

6.6.6.2.3 Test sequences

Table 52 – 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	er_bs01_ep0	er_bs01_ep1	er_bs02_ep0	er_bs02_ep1	er_bs03_ep0	er_bs03_ep1	er_bs04_ep0	er_bs04_ep1	er_bs05_ep0	er_bs05_ep1	er_bs06_ep0	er_bs06_ep1
content	sine sweep	sine sweep	music	music	music	music	music	music	music	music	music	music
Base Layer Bitrate (kbit/s)	16	16	16	16	32	32	32	32	32	32	32	32
Top Bitrate (kbit/s)	40/64	40/64	40/64	40/64	80/128	80/128	80/128	80/128	80/128	80/128	80/128	80/128
Top Layer (n)	24/48	24/48	24/48	24/48	24/48	24/48	24/48	24/48	24/48	24/48	24/48	24/48
number of ES	1	6	25/49	6	2	6	4	6	5	6	3	6
ES Bitrate (kbit/s)	40/64	BL1,BL2 ,6/12, 6/12, 6/12, 6/12	BL,1,1, ..., 1, 1	BL1,BL2 ,6/12, 6/12, 6/12, 6/12	BL,24/48	BL1,BL2 ,12/24,1 2/24,12/ 24,12/24	BL,24(48) ,12(24),1 2(24)	BL1,BL2 ,12/24,1 2/24,12/ 24,12/24	BL,12/24 ,12/24,1 2/24,12/ 24	BL1,BL2 ,12/24,1 2/24,12/ 24,12/24	BL,24/48 ,24/48	BL1,BL2 ,12/24,1 2/24,12/ 24,12/24
number of channel	1	1	1	1	2	2	2	2	2	2	2	2
Intensity									Yes	Yes	Yes	Yes
MS							Yes	Yes	Yes	Yes	Yes	Yes
TNS							Yes	Yes	Yes	Yes	Yes	Yes
epConfig	0	1	0	1	0	1	0	1	0	1	0	1
SBA											Yes	Yes
highestLayer	24/48	24/48	0, 1, 2, ... , 24/48	0, 1, 2, ... , 24/48	0, 24/48	0, 24/48	0, 12/24, 18/36, 24/48	0, 12/24, 18/36, 24/48	0, 6/12, 12/24, 18/36,24 /48	0, 6/12, 12/24, 18/36,24 /48	0, 12/24, 24/48	0, 6/12, 12/24, 18/36,24 /48
Test Procedure	RMS	RMS										

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.

6.6.7 CELP

6.6.7.1 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

6.6.7.2 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]

6.6.7.3 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.

6.6.7.4 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.

6.6.7.5 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 subclause 6.6.1.2.2.1)

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.

6.6.7.6 Descriptions of the audio test bitstreams

Table 53 — 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 54 — 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

6.6.8 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.

6.6.8.1 Compressed data

6.6.8.1.1 Characteristics

6.6.8.1.1.1 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

6.6.8.1.1.2 Bitstream payload

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

LPC_Present

interpolation_flag

gain_indices[1]

6.6.8.1.2 Test procedure

In case that DecoderConfigDescriptor() (see ISO/IEC 14496-1 MPEG4 Systems) is used for MPEG-4 V2 CELP audio decoders, the AudioSpecificConfig must comply with the semantic conditions described below:

6.6.8.1.2.1 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.

6.6.8.1.2.2 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.

6.6.8.1.2.3 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'.

6.6.8.1.2.4 RPE_frame

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

6.6.8.2 Decoders

6.6.8.2.1 Characteristics

6.6.8.2.1.1 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.

6.6.8.2.1.2 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.

6.6.8.2.1.3 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.

6.6.8.2.2 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 55

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 subclause 6.6.1.2.2.1.

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.

6.6.8.2.3 Test sequences

Table 56 – 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 57 – 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).

6.6.9 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.

6.6.9.1 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

6.6.9.2 Audio Access Unit Characteristics

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

6.6.9.3 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, subpart 2, subclause 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, subpart 2, subclause 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 58 — 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.

6.6.9.4 Procedure to Test Bitstream Conformance

6.6.9.4.1 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).

6.6.9.4.2 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.

6.6.9.5 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.

6.6.9.6 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 6 shows the decoder output signal timing for testing decoder conformamce.

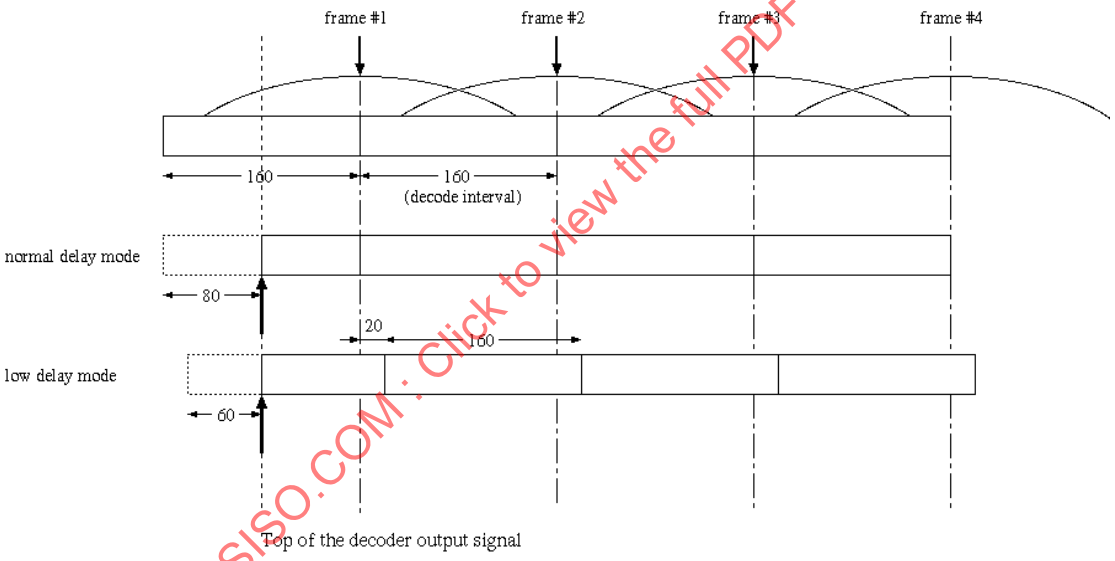


Figure 6 — Decoder output signal timing for testing decoder conformance

6.6.9.6.1 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.

6.6.9.6.1.1 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 4 and Figure 5 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.

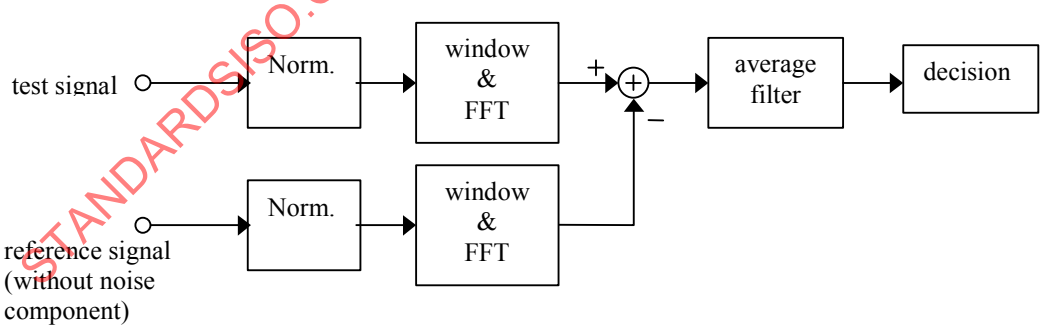


Figure 7 — Block Diagram of the Conformance Testing Procedure (All Voiced bitstream)

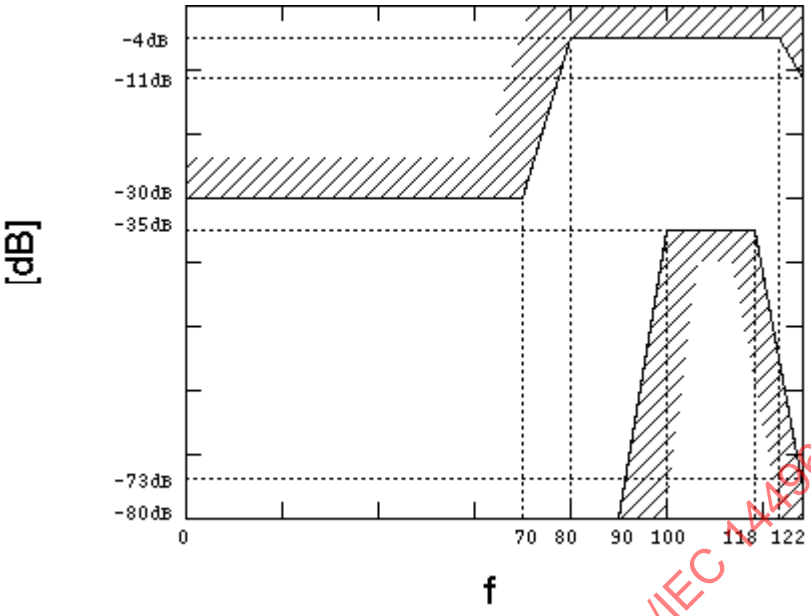


Figure 8 — AccepTable range of differential spectrum (for bitstream HV01)

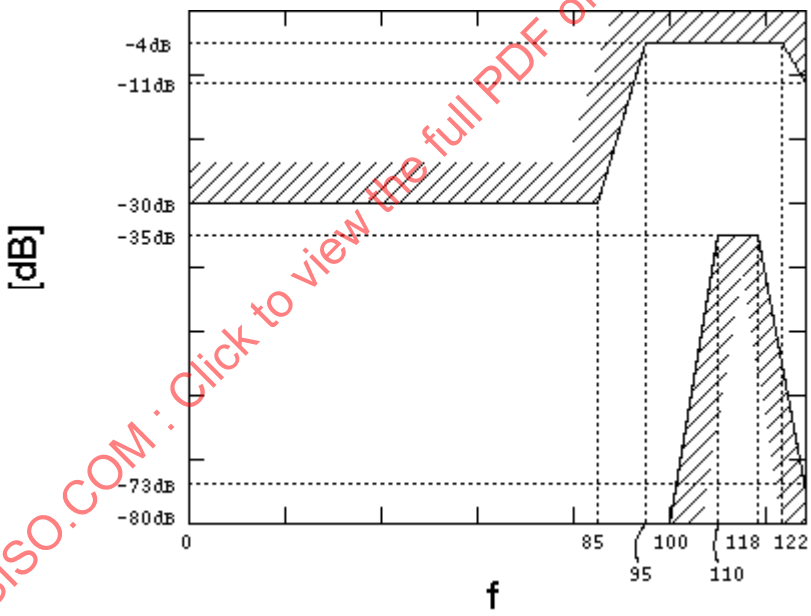


Figure 9 — Acceptable range of differential spectrum (for bitstream HV02)

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");
      }
    }
  }
```

6.6.9.6.1.2 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 subclause 6.6.1.2.2.2, 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

6.6.9.6.2 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 subclause 6.6.1.2.2.2) with random number generators disabled.
- Harmonic phase initialization (verification of phase randomness)

6.6.9.6.2.1 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 subclause 6.6.1.2.2.2) 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

6.6.9.6.2.2 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, subpart 2, subclause 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 10. 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 inTable 59, it can be said that the decoder under the test satisfies the conformance condition.

The above testing procedure is executed using dedicated software.

Table 59 — 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]

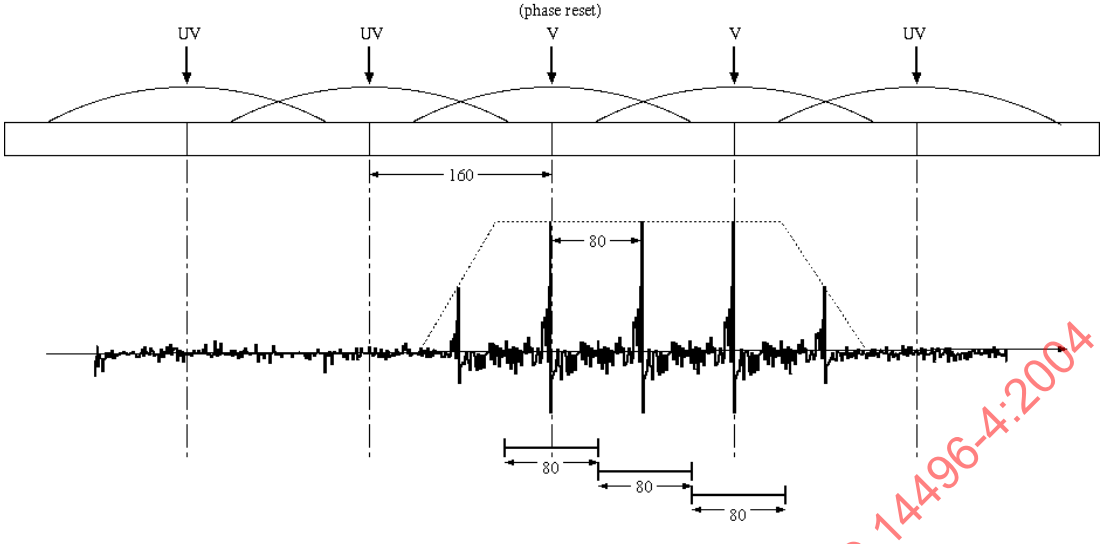


Figure 10— An example decoder output signal and HVXC framing timing

6.6.9.7 Descriptions of the audio test bitstreams

Table 60 — 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 61 — 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

6.6.9.8 Descriptions of the audio reference output signal

Table 62 — 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 63 — 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 64 — 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 65 — 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 66 — 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 67 — 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

6.6.10 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.

6.6.10.1 Compressed Data

6.6.10.1.1 Characteristics

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

6.6.10.1.1.1 AudioSpecificConfig

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

AudioObjectType

samplingFrequencyIndex

channelConfiguration

extensionFlag

epConfig

6.6.10.1.1.2 Bitstream payload

None.

6.6.10.1.1.3 BIFS/AudioSource node

A conform decoder may support any of the modification of the same parameters used in HVXC object type (see subclause 6.6.9.3).

6.6.10.1.2 Test procedure

The AudioSpecificConfig must comply with the semantic conditions described below.

Table 68 – 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.

6.6.10.2 Decoders

6.6.10.2.1 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.

6.6.10.2.2 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 subclause 6.6.9.6.

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.

6.6.10.2.3 Test sequences

Table 69 – 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 70 – ER HVXC Object Type Test Compressed data

File base name	er_hv05_ep 0(1)	er_hv06_ep 0(1)	er_hv07_ep 0(1)	er_hv08_ep 0(1)	er_hv09_ep 0(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 71 – 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

6.6.11 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.

6.6.11.1 Compressed Data

6.6.11.1.1 Characteristics

6.6.11.1.1.1 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 72

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 73

audioObjectType	=27
samplingFrequencyIndex	=0xb
channelConfiguration	=1

Elements of PARAconfig for the ER-HILN object type:

Table 74

PARAmode	=1
extensionFlag	=0

Elements of PARAconfig for the ER-Parametric object type:

Table 75

extensionFlag	=0

Elements of HILNconfig for object type ER-HILN:

Table 76

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 77

HILNsampleRateCode	=11
HILNframeLength	=320
HILNcontMode	≤2

Elements of ErHVXCconfig for object type ER-Parametric: see subclause 6.6.10

The maximum number of HILN extension layers is 7.

6.6.11.1.1.2 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 subclause 1.5.2.3 of ISO/IEC 14496-3:2001. The calculation of PCU values for HILN is described in subclause 1.5.2.2 of ISO/IEC 14496-3:2001. 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 78

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 79

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

6.6.11.1.2 Test procedure

A conforming compressed data must comply to the restrictions specified in subclause 6.6.1.1.1. 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.

6.6.11.2 Decoders

6.6.11.2.1 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.

6.6.11.2.2 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 subclause 6.6.11.2.3 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 subclauses 6.6.11.2.2.3 to 6.6.11.2.2.5, 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.

6.6.11.2.2.1 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 subclause 6.6.1.2.2.1.

6.6.11.2.2.2 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 14496-4:2000).

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.

6.6.11.2.2.3 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 subclause 6.6.11.2.2.6. 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.

6.6.11.2.2.4 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 subclause 6.6.11.2.2.6. 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
```

6.6.11.2.2.5 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 subclause 6.6.11.2.2.3. 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 subclause 6.6.11.2.2.6. 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
```

6.6.11.2.2.6 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*framelen*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=arc(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<nfrmes*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);
}
```

6.6.11.2.3 Test sequences

Table 80 through Table 83 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 class	see subclause 6.6.11.2.2.1 or subclause 6.6.11.2.2.2 according to selected accuracy
noise gen	see subclause 6.6.11.2.3 [1])
noise env	see subclause 6.6.11.2.4 [1])
start phase	see subclause 6.6.11.2.5 [1])
det+noise subtracted signal	first the reference waveform (containing only the deterministic components) is subtracted and then test procedure described in subclause 6.6.11.2.3 is applied to the residual 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 83.

Table 80 – 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 81 – 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 82 – 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 83 – 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

6.6.12 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 11 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 11 are distinguished in the Table 84.

Table 84 — 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

6.6.12.1 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.

6.6.12.2 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

6.6.12.3 Procedure to Test Bitstream Conformance

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

6.6.12.4 Decoder Characteristics

Decoder conformance test will be applied to several interfaces identified in the Table 84. 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.

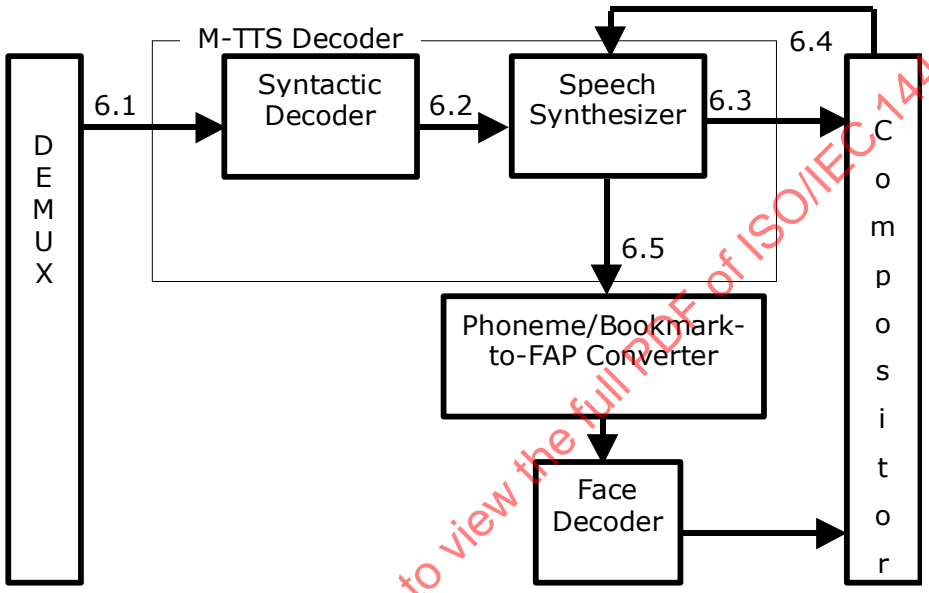


Figure 11 — MPEG-4 Audio TTS decoder architecture. (Text-to-Speech in ISO/IEC 14496-3)

6.6.12.5 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 85. 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 85. 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.

6.6.12.6 Descriptions of the test bitstreams

ISO/IEC 14496-3 (MPEG-4) audio test bitstreams are suggested as follows:

Table 85 — 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.

6.6.13 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.

6.6.13.1 Procedure to Test Bitstream Conformance

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

6.6.13.1.1 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.

6.6.13.1.2 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 subclause 5.5.2 of subpart 5 of ISO/IEC 14496-3.

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

6.6.13.2 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.

6.6.14 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)

6.6.14.1 Procedure to Test Decoder Conformance

6.6.14.1.1 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.

6.6.14.1.2 Audio Access Unit Characteristics

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

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

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

6.6.14.2 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.

6.6.15 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.

6.6.15.1 DecoderSpecificInfo Characteristics

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

a) **orchestra** block

6.6.15.2 Audio Access Unit Characteristics

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

6.6.15.3 Procedure to Test Bitstream Conformance

6.6.15.3.1 DecoderSpecificInfo Characteristics

AudioObjectType: Shall be encoded with the value 16

SamplingFrequencyIndex: Shall be encoded with the following values:

Table 86

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 87

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 88

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.

6.6.15.3.2 Audio Access Unit Characteristics

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

6.6.15.4 Decoder Characteristics

All signal variables in SAOL shall be represented by a 32-bit floating-point value as defined in ISO/IEC 14496-3, subpart 5, subclause 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 B. 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 89, 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 89 — 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 B, 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 subclauses 7.3.3.7 and 7.3.3.8 of ISO/IEC 14496-3 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 subclause 6.8.7 and Table 90, where these values are used.

Table 90 — 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

6.6.15.5 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 subclauses 9.8 and 10.4 of ISO/IEC 14496-3 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 the subclause 6.6.1.2.2.1. 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 bitstreams SY001 through SY004.

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 89. See also subclause 6.6.15.4 and subclauses 7.3.3.7 and 7.3.3.8 of ISO/IEC 14496-3 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 89. 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 subclause 6.6.15.4. 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 subclause 6.6.15.6. 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 12.

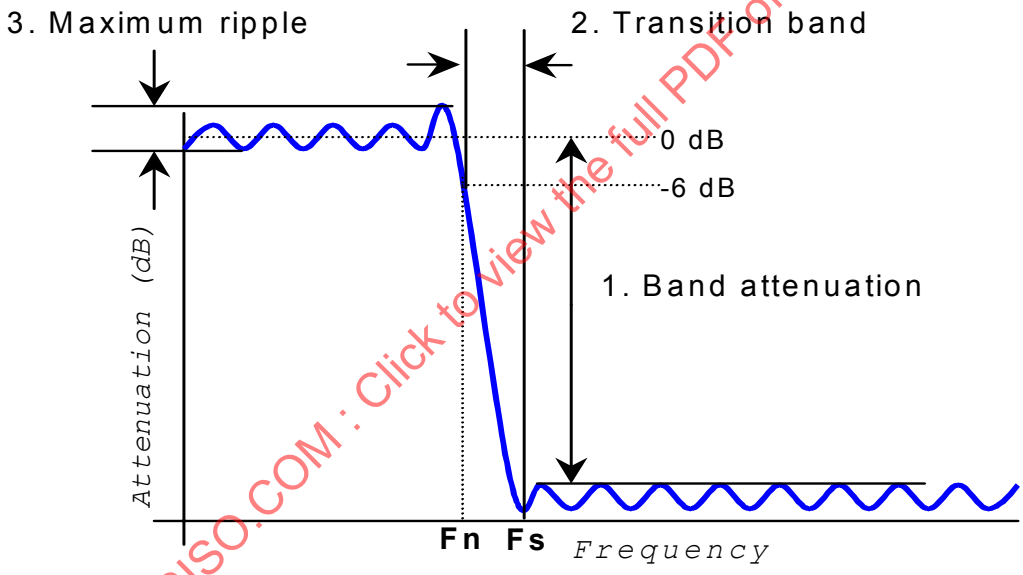


Figure 12

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 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."

6.6.15.6 Descriptions of Conformance Bitstreams

All conformance bitstreams whose memory requirements and processing level, as indicated in Table 91 and Table 92, 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 (subclause 9.4 in ISO/IEC 14496-3 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 subclause 6.6.15.5 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.

Table 91 — 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 92 — 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

6.6.16 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.

6.6.16.1 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.

6.6.16.2 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.

6.6.16.3 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.

6.6.16.3.1 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.

6.6.16.3.2 Audio Access Unit Characteristics

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

6.6.16.4 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

6.6.16.5 Descriptions of Conformance Bitstreams

See sections on the following audio object types:

- MIDI
- Wavetable synthesis
- Algorithmic synthesis and AudioFX

6.7 Audio EP tool

6.7.1 Compressed data

6.7.1.1 Characteristics

Encoders may apply restrictions to the following parameters of the compressed data[2]:

6.7.1.1.1 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

termination_switch

interleave_switch

class_optional

number_of_bits_for_length

class_length

class_rate

class_crclen

class_reordered_output

class_output_order

2 With respect to the AudioSpecificConfig, only parameters of ErrorProtectionSpecificConfig are mentioned. For all other parameters please refer to the appropriate subclause 6.6.

header_protection

header_rate

header_crclen

rs_fec_capability

6.7.1.1.2 Bitstream payload

None.

6.7.1.2 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.

6.7.1.2.1 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.

6.7.1.2.2 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.

6.7.2 Decoders

6.7.2.1 Characteristics

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

Table 93 – 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

6.7.2.2 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 94.

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 subclause 6.3: The output of a conforming decoder shall be similar independent of the used epConfig setting

Table 94 – 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	0,0,1,0,1,0,1	8,10,10,14,10,10,0	-	0,0,0,0,0,0,0	5,6,7,0,0,8,9,0	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	-	1,0,1,0,1,1,0,1,0	-,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	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	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	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	1,0,0,0,0,0,0,0,0	2,2,1,0,3,2,1,0,1	0,0,1,0,0,0,1,0,0	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	1,1,0,1	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			-	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	-	2,2,0,0,0,1,0,0,0	-,-,1,0,-,0,1,-	-	0,0,1,0,0,1,0,0,0	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	-	0,0,0,1,0,0,0	8,10,10,14,10,10,0	-	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	-	-
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	-	-

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
		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	8,7,6,5,0,4,3,2,1,0	0	-	-
		0	2	-	-	-	-	-	-	2	0,0	0,0	0,0	-	0,0	0,0	-	0,0	-	-	6,6	8,8	0	-	0	-	-
		1	3	1	6	1	0	1	1	9	0,0,0,0,0,0,0,0,0	0,0,0,0,0,0,0,0,0	0,0,0,0,0,0,0,0,0	-	0,1,2,1,0,0,1,1,0	1,-,-,-,-,1,0,-,-0	3,3,1,1,0,1,1,3,2	0,0,0,1,0,0,0,1,0	8,10,10,10,14,10,10,10,14	-	80,400	6,6	8,8	0	-	0	-	-
		2	3	1	6	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	-	0,1,2,1,0,0,1,1,0	-,-,-,-,-,-,-,-,-	-	0,0,0,1,0,0,0,1,0	8,10,10,10,14,10,10,10,14	-	80,400	6,6	8,8	0	-	0	-	-
		-	2	-	-	-	-	-	-	7	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	-	2,1,1,2,2,2,1	-,-,-,-,-,-,-,-,-	-	0,0,0,1,0,0,0,1,0	8,10,10,10,14,10,10,10,14	-	128,896	6,0	8,0	0	-	0	-	-
		0	2	-	3	1	1	1	1	2	0,0,0,0,0,0,0,0,0	0,0,0,0,0,0,0,0,0	0,0,0,0,0,0,0,0,0	-	0,0	0,0	-	0,0	8,10,10,14,10,10,14	-	136,200	6,0	8,0	0	-	0	-	-
		1	2	-	3	1	2	1	1	2	0,0,0,0,0,0,0,0,0	0,0,0,0,0,0,0,0,0	0,0,0,0,0,0,0,0,0	-	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	0,0,0,0,0,0,0,0,0	0,0,0,0,0,0,0,0,0	-	0,0	0,0	-	0,0	-	-	64,272	2,0	8,0	0	-	0	-	-
er_ce00			2	-	-	4	0 / 1 / 1 / 0 /	1 / 1 / 1 / 1 /	1 / 1 / 1 / 1 /	1 / 1 / 5 / 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	-	-	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		-	2	-	-	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	2	-	-	4	0 / 1 / 1 / 0 /	1 / 1 / 1 / 1 /	1 / 1 / 1 / 1 /	1 / 1 / 5 / 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	-	-	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	2	-	-	2	0 / 0	1 / 1	1 / 1	0 / 1	-/ 0	-/ 0	-/ 0	-	-/ 0	-/ 0	-	-/ 0	-	-	0 / 60	-/ 0	-/ 8	0	-	0	-	-
		2	2	-	-	2	0 / 0	1 / 1	1 / 1	0 / 1	-/ 0	-/ 0	-/ 0	-	-/ 0	-/ 0	-	-/ 0	-	-	0 / 60	-/ 0	-/ 8	0	-	0	-	-
		3	2	-	-	3	0 / 0 / 0	1 / 1 / 1	1 / 1 / 1	0 / 1 / 1	-/ 0 / 0	-/ 0 / 0	-/ 0 / 0	-	-/ 0 / 0	-/ 0 / 0	-	-/ 0 / 0	-	-	0 / 320 / 21	-/ 10 / 5	-/ 8	0	-	0	-	-
er_ce09		-	2	-	-	5	0 / 1 / 1 / 0 /	1 / 1 / 1 / 1 /	1 / 1 / 1 / 1 /	1 / 1 / 5 / 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	-	-	2 / 21,14,19,39,7 / 6,8,9,26,75 / 40	8 / 8,6,4,2,0 / 8 / 8	5 / 5,6,7,8,0 / 5 / 5	0	-	0	-	-
er_ce13		-	2	-	-	2	1 / 1	1 / 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	34,18,26,21,6 / 19,12,16,8,61	-	3 /	8,6,4,2,0 / 8,6,4,2,0	5,6,7,8,0 / 5,6,7,8,0	0	-	0	-	-

header_crclen	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
header_rate	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
header_protection	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
class_output_order	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
class_reordered_output	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
class_crclen	5 / 5,6,7,8,0 / 5 / 5	5 / 5,6,7,8,0 / 5,6,7,8,0 / 5 / 5	6,1,1,0	6,6,1,1,0	6,1,1,0	6,6,1,1,0	7,10,10, 10,10,10	7,10,10, 10,10,10	7,10	10,10,10,10	7,10	10,10	10	10	7,10	10	10	10	10	7,10	10,10	10,10
class_rate	8 / 8,6,4,2,0 / 8 / 8	8 / 8,6,4,2,0 / 8,6,4,2,0 / 8 / 8	0,0,0,0	0,0,0,0,0	8,0,0,0	8,0,0,0,0	20,16,8, 0,0,0	20,16,8, 0,0,0	20,16	8,0,0,0	20,16	8,0	0	0	20,16	8	0	0	0	20,16	8,0	8,0
class_length	2 / 50,30,42,29,1 31 / 40 / 8	2 / 60,36,50,33,1 63 / 45,30,40,20,1 61 / 40 / 8	22,4,4,10	33,22,4,4,17	22,4,4,10	33,22,4,4,17	-	-	-		-	-	-	-	-	-	-	-	-	-	-	-
number_of_bits_for_length	-	-	-	-	-	-	6,9,9,9,9,0	6,9,9,9,9,0	6,0	9,9,9,0	6,0	9,0	0	0	6,9	0	0	0	0	6,0	9,0	9,0
class_optional	0 / 0,0,0,0,0 / 0 / 0	0 / 0,0,0,0,0 / 0,0,0,0,0 / 0 / 0	0,0,0,0	0,0,0,0,0	0,0,0,0	0,0,0,0,0	0,0,0,0,0,0	0,0,0,0,0,0	0,0	0,0,0,0	0,0	0,0	0	0	0,0	0	0	0	0	0,0	0,0	0,0
interleave_switch	-	-	-	-	2,3,3,3	2,3,3,3,3	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
termination_switch	0 / 0,0,0,0,0 / 0 / 0	0 / 0,0,0,0,0 / 0,0,0,0,0 / 0 / 0	0,0,0,0	0,0,0,0,0	0,0,0,0	0,0,0,0,0	0,0,0,0,0,0	0,0,0,0,0,0	0,0	0,0,0,0	0,0	0,0	0	0	0,0	0	0	0	0	0,0	0,0	0,0
fec_type	0 / 0,0,0,0,0 / 0 / 0	0 / 0,0,0,0,0 / 0,0,0,0,0 / 0 / 0	0,0,0,0	0,0,0,0,0	0,0,0,0	0,0,0,0,0	0,0,0,0,0,0	0,0,0,0,0,0	0,0	0,0,0,0	0,0	0,0	0	0	0,0	0	0	0	0	0,0	0,0	0,0
concatenated_flag	-	-	1,0,0,1	1,1,0,0,1	1,0,0,1	1,1,0,0,1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
crc_len_escape	0 / 0,0,0,0,0 / 0 / 0	0 / 0,0,0,0,0 / 0,0,0,0,0 / 0 / 0	0,0,0,0	0,0,0,0,0	0,0,0,0	0,0,0,0,0	0,0,0,0,0,0	0,0,0,0,0,0	0,0	0,0,0,0	0,0	0,0	0	0	0,0	0	0	0	0	0,0	0,0	0,0
rate_escape	0 / 0,0,0,0,0 / 0 / 0	0 / 0,0,0,0,0 / 0,0,0,0,0 / 0 / 0	0,0,0,0	0,0,0,0,0	0,0,0,0	0,0,0,0,0	0,0,0,0,0,0	0,0,0,0,0,0	0,0	0,0,0,0	0,0	0,0	0	0	0,0	0	0	0	0	0,0	0,0	0,0
length_escape	0 / 0,0,0,0,0 / 0 / 0	0 / 0,0,0,0,0 / 0,0,0,0,0 / 0 / 0	0,0,0,0	0,0,0,0,0	0,0,0,0	0,0,0,0,0	1,1,1,1,1,1	1,1,1,1,1,1	1,1	1,1,1,1	1,1	1,1	1	1	1,1	1	1	1	1	1,1	1,1	1,1
number_of_class	1 / 5 / 1 / 1	1 / 5 / 5 / 1 / 1	4	5	4	5	6	6	2	4	2	2	1	1	2	1	1	1	1	2	2	2
number_of_concatenated_frame	1 / 1 / 1 / 1	1 / 1 / 1 / 1 / 1	2	2	2	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
bit_stuffing	1 / 1 / 1 / 1	1 / 1 / 1 / 1 / 1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
interleave_type	0 / 1 / 0 / 0	0 / 1 / 1 / 0 / 0	0	0	2	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
number_of_predefinition_set	4	5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
SamplingFrequencyIndex (if not specified: same as for epConfig=0,1)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
directMaping	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
epConfig	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
layer	-	-	-	-	-	-	-	-	0	1	0	1	2	3	0	1	2	3	4	0	1	2
epSetup			crc	crc	fec	fec																
coreSetup	er_ce15	er_ce18	er_hv03	er_hv04	er_hv03	er_hv04	er_bs01	er_bs02	er_bs03		er_bs04				er_bs05					er_bs06		

header_crclen	-	-	-	-	-	-	-	-
header_rate	-	-	-	-	-	-	-	-
header_protection	0	0	0	0	0	0	0	0
class_output_order	-	-	-	-	-	-	-	-
class_reordered_output	0	0	0	0	0	0	0	0
class_crclen	6,9,12,12,4	6,9,12,12,4	6,9,12,12,4	6,9,12,12,4	6,9,12,12,4	6,9,12,12,4	6,9,12,12,4	6,9,12,12,4
class_rate	0,0,0,0,0	0,0,0,0,0	0,0,0,0,0	0,0,0,0,0	8,4,2,0,0	8,4,2,0,0	8,4,2,0,0	8,4,2,0,0
class_length	-	-	-	-	-	-	-	-
number_of_bits_for_length	6,8,12,12,0	6,8,12,12,0	6,8,12,12,0	6,8,12,12,0	6,8,12,12,0	6,8,12,12,0	6,8,12,12,0	6,8,12,12,0
class_optional	0,0,0,0,1	0,0,0,0,1	0,0,0,0,1	0,0,0,0,1	0,0,0,0,1	0,0,0,0,1	0,0,0,0,1	0,0,0,0,1
interleave_switch	-	-	-	-	-	-	-	-
termination_switch	0,0,0,0,0	0,0,0,0,0	0,0,0,0,0	0,0,0,0,0	0,0,0,0,0	0,0,0,0,0	0,0,0,0,0	0,0,0,0,0
fec_type	0,0,0,0,0	0,0,0,0,0	0,0,0,0,0	0,0,0,0,0	0,0,0,0,0	0,0,0,0,0	0,0,0,0,0	0,0,0,0,0
concatenated_flag	-	-	-	-	-	-	-	-
crc_len_escape	0,0,0,0,0	0,0,0,0,0	0,0,0,0,0	0,0,0,0,0	0,0,0,0,0	0,0,0,0,0	0,0,0,0,0	0,0,0,0,0
rate_escape	0,0,0,0,0	0,0,0,0,0	0,0,0,0,0	0,0,0,0,0	0,0,0,0,0	0,0,0,0,0	0,0,0,0,0	0,0,0,0,0
length_escape	1,1,1,1,1	1,1,1,1,1	1,1,1,1,1	1,1,1,1,1	1,1,1,1,1	1,1,1,1,1	1,1,1,1,1	1,1,1,1,1
number_of_class	5	5	5	5	5	5	5	5
number_of_concatenated_frame	1	1	1	1	1	1	1	1
bit_stuffing	1	1	1	1	1	1	1	1
interleave_type	0	0	0	0	1	1	1	1
number_of_predefinition_set	1	1	1	1	1	1	1	1
SamplingFrequencyIndex (if not specified: same as for epConfig=0,1)	-	-	-	-	-	-	-	-
directMaping	1	1	1	1	1	1	1	1
epConfig	3	3	3	3	3	3	3	3
layer	-	-	-	-	-	-	-	-
epSetup	crc	crc	crc	crc	fec	fec	fec	fec
coreSetup	er_hi26	er_hi27	er_hi28	er_hi29	er_hi26	er_hi27	er_hi28	er_hi29

6.8 Audio Composition

6.8.1 Introduction

This part defines the conformance of audio composition using AudioBIFS nodes as defined in ISO/IEC 14496-1 (Systems). 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).

6.8.1.1 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 95 — 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 = ***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 subclause 5.5.2 of ISO/IEC 14496-3: Complexity Units
- Spatializable Object types: number of possible spatialized channels
- AudioFX: see subclause 6.6.15
- Reaction Time of a BIFS field update: the maximum time in msec. until the changes is audible.

6.8.1.2 Levels for Systems Audio Scene Graph Profile

Following these considerations, audio composition Levels are defined in the form of the following table:

Table 96 — 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 90)	Low Complexity (Table 90)	Medium Complexity (Table 90)	High Complexity (Table 90)
Spatialization	0	4	16	32

6.8.1.3 Composition Unit Inputs

Table 97

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.

6.8.1.4 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 in subclauses 9.4.2.82 and 9.4.2.83 of ISO/IEC 14496-1.

Audio composition is tested for Conformance as described in the following subclauses 6.8.3 to 6.8.7. Test scenes are defined using composition units described in subclause 6.8.1.3 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 subclause 6.8.1.3, 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.

6.8.2 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.

6.8.2.1 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.

6.8.2.2 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.

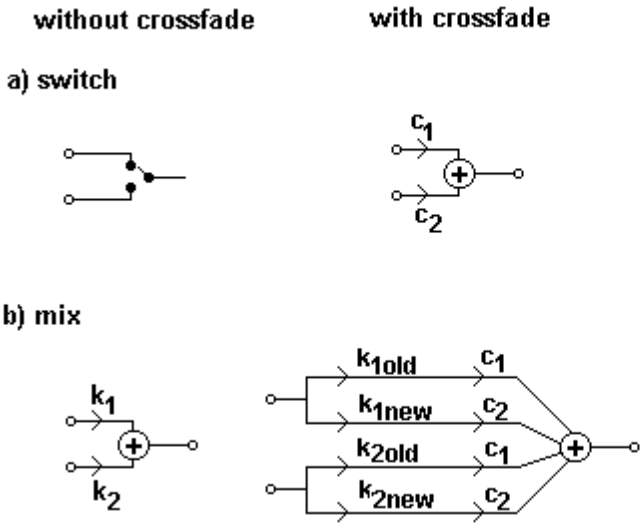


Figure 13 — Click free switch and mix

The above explained cross fade applies to the nodes **AudioSource**, **AudioMix**, **AudioSwitch**, and **AudioClip**.

6.8.2.3 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.

6.8.3 AudioSource and Sound2D

6.8.3.1 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, subclause 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, subclause 5.3).

6.8.3.2 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 (subclause 6.6.1.2.2.1). 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.

6.8.3.3 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 14 — 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 14, 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 98 — AudioBIFS Test Bitstream

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

6.8.4 AudioSource and Sound

6.8.4.1 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, subclause 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, subclause 5.3).

6.8.4.2 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 (subclause 6.6.1.2.2.2). 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.

6.8.4.3 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. Height 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 99 — AudioBIFS Test Bitstream

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

6.8.5 AudioSwitch

See also subclause 6.8.2.2. Conformance Test of the AudioSwitch node is not required for decoders at Level 1.

6.8.5.1 BIFS fields Characteristic

None.

6.8.5.2 Procedure to Test Audio Node

Testing the AudioSwitch Scene 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.

6.8.5.3 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.

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.

Table 100

File Name	AB031
Content	BIFS
Bitstream 1 from a source (url)	CU1
Bitstream 2 from a source (url)	CU1

6.8.6 AudioMix and Sampling Rate Conversion

See also subclause 6.8.2.2.

6.8.6.1 BIFS fields Characteristic

None.

6.8.6.2 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 the subclause “Sampling Rate conversion.”(subclause 6.8.2.3) 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.

6.8.6.3 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 101

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

6.8.7 AudioFX

See also subclause 6.6.15

6.8.7.1 BIFS fields Characteristic

Restrictions on field values.

6.8.7.2 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 (subclause 6.6.1.2.2.2). This test only verifies the computational accuracy of an implementation.

6.8.7.3 Audio BIFS Test Scenes

AB101 One AudioSource node connected to one AudioFX node with Stripe orchestra and Score and to a Sound2D node with default fields, using CU1

AB102 One AudioSource node connected to one AudioFX node with Stripe orchestra and Score and to a Sound2D node with default fields, using CU4

For sequences AB101 and AB102 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. The electronic attachment to this part of ISO/IEC 14496 also provides reference output.

Table 102 — AudioBIFS Test Bitstream

File Name	AB101	AB102
Content	Stripe	Stripe
Orchestra definition (orch)	Stripe	Stripe
Score definition (score)	Stripe	Stripe
Bitstream 1 from a source (url)	CU1	CU4

6.9 MPEG-4 audio transport stream

This clause defines conformance points and conformance testing procedures for MPEG-4 Audio decoders without using MPEG-4 Systems. Such decoders have a mechanism to receive MPEG-4 Audio Transport Stream as shown in Figure 15. The mechanism consists of a multiplex layer and a synchronization layer. The multiplex layer (Low-overhead MPEG-4 Audio Transport Multiplex: LATM) manages multiplexing of several MPEG-4 Audio payloads and AudioSpecificConfig elements. The synchronization layer specifies a self-synchronized syntax of the MPEG-4 Audio transport stream.

The conformance points for the LATM-based decoder are defined at the output waveforms of the audio decoder. The conformance testing procedures defined in this clause are applied to verfiy multiplex and synchronization processes. The output waveforms are evaluated in accordance with the conformance testing procedure, which depends on Audio Object Type.

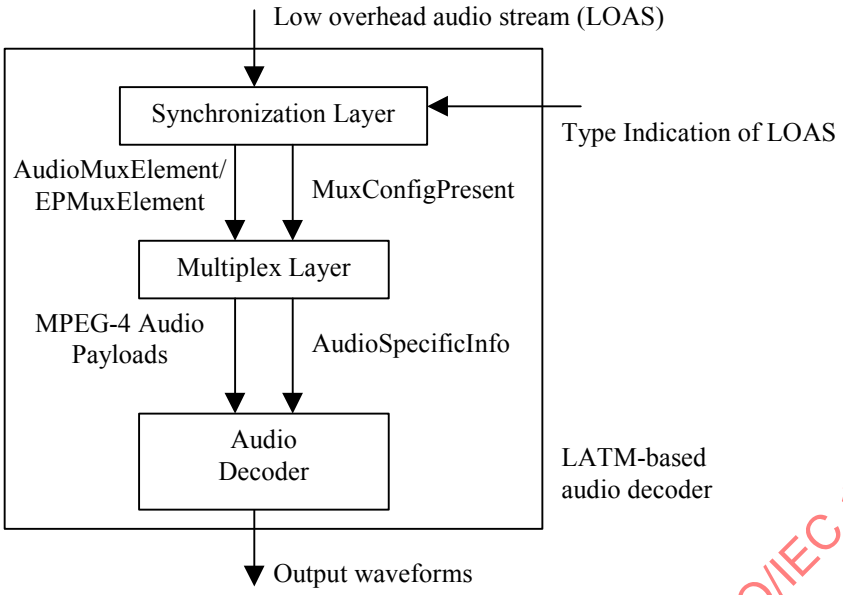


Figure 15 – Audio Conformance Points for LATM-based decoders

6.9.1 Compressed Data

6.9.1.1 Synchronization Layer

The following restrictions apply to the AudioSyncStream, the EP_AudioSyncStream or the audioPointerStream:

audioMuxLengthBytesLast: shall not be encoded with the value 0.

audioMuxLengthBytes: shall not be encoded with the value 0.

audioMuxElementsStartPointer: shall not be encoded with the value 0.

6.9.1.2 Multiplex Layer

The following restrictions apply to the multiplexed element:

CELPframeLengthTableIndex: In the core layer of CELP scalable compressed data, this field shall be encoded with the value less than 62. For the enhancement layer, this field shall be encoded with the value less than 20.

MuxSlotLengthCoded[]: When frameLengthType equals 3, this field shall be encoded with the value less than 2.

For AudioSpecificConfig and audio payload data, the restriction defined for the corresponding Audio Object type shall be applied. The **audioObjectType** element in AudioSpecificConfig shall not be encoded with the values from 12 to 16.

6.9.2 Decoders

To test LATM audio decoders, a number of test sequences are supplied. This test only verifies the functionality of an implementation of the LATM audio decoder.

For a supplied test sequence, testing can be done using the procedures defined for the corresponding Audio Object type.

6.9.2.1 Test sequences

The test compressed data in the LOAS format is generated from the MP4FF compressed data which is supplied for each Audio Object type. The corresponding reference waveforms are also used in this evaluation. Therefore, the LOAS test compressed data only is supplied.

Table 103

File base name	er_ce02_ep0	er_ce0203_ep0	er_ce06a_ep0	er_ce06b_ep0
Original Compressed Data	er_ce02_ep0	er_ce02_ep0 er_ce03_ep0	er_ce06_ep0	er_ce06_ep0
Corresponding Reference Waveforms	er_ce02	er_ce02 er_ce03	er_ce06_lay0 er_ce06_lay1 er_ce06_lay2 er_ce06_lay3	er_ce06_lay0 er_ce06_lay1 er_ce06_lay2 er_ce06_lay3
LOAS Type	AudioSyncStream	AudioSyncStream	AudioSyncStream	AudioSyncStream
allStreamSameTimeFraming	1	1	1	0
numSubFrames	3	1	1	1
numProgram	1	2	1	1
numLayer	1	1	4	4

6.10 Upstream

6.10.1 Compressed data

6.10.1.1 Characteristics

6.10.1.1.1 AudioSpecificConfig

There is constraint for the value of AudioSpecificConfig. An encoder may apply restrictions to the following parameters of the AudioSpecificConfig:

AudioObjectType

6.10.1.1.2 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:

upstreamType

6.10.1.2 Test procedure

6.10.1.2.1 AudioSpecificConfig

The following restrictions apply to AudioSpecificConfig:

AudioObjectType: Shall be encoded with the value decoded from the down stream.

6.10.1.2.2 Audio Access Units

upstreamType: shall be smaller than or equal to 3. Shall not be 1 if AudioObjectType is not 22.

numOfLayer: shall be larger than 0.

numOfSubFrame: shall be larger than 0.

6.10.2 Decoders

No decoder conformance is specified, because the upstream decoder is part of the encoder, which is not standardized.

6.11 Advanced Audio BIFS nodes

6.11.1 Introduction

Advanced AudioBIFS nodes are used for adding advanced 3-D audio processing functionalities for Virtual Reality rendering purposes in 3-D scenes.

6.11.2 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

6.11.3 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).

6.11.4 Physical Approach

This clause describes the conformance testing for the rendering (audio output) of Advanced AudioBIFS nodes (physical approach, as described in subclause 9.2.2.13.4 of ISO/IEC 14496-1.

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.

6.11.4.1 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, ISO/IEC 14496-1 subclause 9.4.2.92)

PositionInterpolator (See, ISO/IEC 14496-1 subclause 9.4.2.73) and/or OrientationInterpolator (ISO/IEC 14496-1 subclause 9.4.2.66)

ROUTE syntax (See, ISO/IEC 14496-1 subclause 9.2.2.8.1.4 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.

6.11.4.2 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 the subclauses 6.11.4.3, 6.11.4.4, and 6.11.4.5 the testing procedures are described for **DirectiveSound**, **AcousticScene**, and **AcousticMaterial**, respectively, so that all the functionalities of these three nodes are covered.

6.11.4.3 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

6.11.4.3.1 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.

6.11.4.3.1.1 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.

6.11.4.3.1.2 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.

6.11.4.3.2 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.

6.11.4.3.2.1 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.

6.11.4.3.2.2 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.

6.11.4.3.3 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.

6.11.4.3.3.1 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.

6.11.4.3.3.2 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 subclause 6.11.4.3.4.1, 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.

6.11.4.3.4 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.

6.11.4.3.4.1 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.

6.11.4.3.4.2 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 subclause 6.11.4.3.4. 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.
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.

6.11.4.3.5 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.

6.11.4.3.5.1 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.

6.11.4.3.5.2 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.

6.11.4.4 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

6.11.4.4.1 Testing of late reverberation

Late reverberation properties are defined by the fields reverbTime, reverbFreq, reverbLevel, and reverbDelay.

6.11.4.4.1.1 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.

6.11.4.4.1.2 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.

6.11.4.4.2 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.

6.11.4.4.2.1 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.

6.11.4.4.2.2 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.

6.11.4.5 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.

6.11.4.5.1 Testing of reflectivity

6.11.4.5.1.1 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).

6.11.4.5.1.2 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 subclause 6.11.4.3.4, 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 subclause 6.11.4.3.3. 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.

6.11.4.5.2 Testing of sound obstruction

6.11.4.5.2.1 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 subclause 6.11.4.3.4, and the attenuation should match that defined by the equation in subclause 6.11.4.3.3.

6.11.4.5.2.2 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.

6.11.4.5.3 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.

6.11.5 Perceptual Approach

This clause describes the conformance testing for the rendering (audio output) of Advanced AudioBIFS nodes (perceptual approach), as described in subclause 9.2.2.13.4 of ISO/IEC 14496-1. 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.

6.11.5.1 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, ISO/IEC 14496-1 subclause 9.4.2.92)

PositionInterpolator (See, ISO/IEC 14496-1 subclause 9.4.2.73) and/or OrientationInterpolator (ISO/IEC 14496-1 sublause 9.4.2.66)

ROUTE syntax (See, ISO/IEC 14496-1 subclause 9.2.2.8.1.4 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.

6.11.5.2 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 the subclauses 6.11.5.3 and 6.11.5.4 the testing procedures are described for **DirectiveSound**, and **PerceptualParameters** respectively.

6.11.5.3 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).

6.11.5.3.1 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.

6.11.5.3.1.1 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.

6.11.5.3.1.2 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.

6.11.5.3.2 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.

6.11.5.3.2.1 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.

6.11.5.3.2.2 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.

6.11.5.3.3 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.

6.11.5.3.3.1 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.

6.11.5.3.3.2 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 subclause 9.4.2.78.2.2 of ISO/IEC 14496-1 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.

6.11.5.3.4 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.

6.11.5.3.4.1 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.

6.11.5.3.4.2 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 subclause 6.11.5.3.4 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.

6.11.5.3.5 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.

6.11.5.3.5.1 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.

6.11.5.3.5.2 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.

6.11.5.4 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

6.11.5.4.1 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 subclauses 9.4.2.78.2, 9.4.2.78.2.1 and 9.4.2.78.2.2 of ISO/IEC 14496-1.

6.11.5.4.1.1 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).

6.11.5.4.1.2 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 subclause 9.4.2.78.2.1 and 9.4.2.78.2.2 of ISO/IEC 14496-1. 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 subclause 9.4.2.78.2.1 and 9.4.2.78.2.2 of ISO/IEC 14496-1 . 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.

6.11.5.4.2 Testing of frequency-dependent effects

This subclause describes testing of **sourceWarmth**, **sourceBrilliance**, **heaviness** and **liveness**.

6.11.5.4.2.1 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.

6.11.5.4.2.2 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 subclause 9.4.2.78.2.1 and 9.4.2.78.2.2 of ISO/IEC 14496-1 . 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 subclause 9.4.2.78.2.1 and 9.4.2.78.2.2 of ISO/IEC 14496-1 . The reverberation time at 1000 Hz, 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 seconds 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 seconds with default values, 5 seconds with heaviness=10.0 and liveness =0.1 and 5 seconds 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.

6.11.5.4.3 Testing of InputFilterGains and directFilterGains

This subclause describes testing of **directFilterGains** and **inputFiltergains** fields.

6.11.5.4.3.1 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.

6.11.5.4.3.2 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 dB.
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 dB.
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 dB 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 dB at frequency freqLow and reduced by 20 dB at frequency freqHigh.
AABper71-72	Same procedure and field values as for AABper69 and AABper70, but with input sound CU4_AAB_Px.

6.11.5.4.4 Testing of omnidirectivity

This subclause describes testing of **omniDirectivity** field.

6.11.5.4.4.1 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.

6.11.5.4.4.2 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-frequency pairs in the omniDirectivity field.
AABper76	Same procedure and field values as for AABper75 but with input sound CU4_AAB_Px.

6.12 Conformance test sequence assignment to profiles and levels

A test sequence, belonging to a certain profile@level is marked by an X. Restrictions, as far as sampling frequency is concerned, are marked by ≥y or <y. This means, that there are sequences within this set of test sequences belonging to more than one level at this profile. y stands for the limiting sample frequency index. For example ≥3 for a certain set of test sequences means that all sequences of this set with a sample frequency index higher or equal to 3 belong to this profile@level.

6.12.1 Audio

Table 104

Object type	sequence name	Main Audio Profile				Scalable Audio Profile				Speech Profile		Synthetic Audio Profile			High Quality Audio Profile								Low Delay Audio Profile								Natural Audio Profile				Mobile Audio Internetworking Profile					
	Level	1	2	3	4	1	2	3	4	1	2	1	2	3	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	1	2	3	4	1	2	3	4	5	6
AAC LC	al00	X	X	X	X	≥6	≥6	≥3	≥3						≥7	≥3	≥3	≥3	≥7	≥3	≥3	≥3									≥3	X	≥3	X						
	al01	X	X	X	X	≥6	≥6	≥3	≥3						≥7	≥3	≥3	≥3	≥7	≥3	≥3	≥3									≥3	X	≥3	X						
	al02	X	X	X	X	≥6	≥6	≥3	≥3						≥7	≥3	≥3	≥3	≥7	≥3	≥3	≥3									≥3	X	≥3	X						
	al03	X	X	X	X	≥6	≥6	≥3	≥3						≥7	≥3	≥3	≥3	≥7	≥3	≥3	≥3									≥3	X	≥3	X						
	al04	X	X	X	X	≥6	≥6	≥3	≥3						≥7	≥3	≥3	≥3	≥7	≥3	≥3	≥3									≥3	X	≥3	X						
	al05	X	X	X	X		≥6	≥3	≥3						≥7	≥3	≥3	≥3	≥7	≥3	≥3	≥3									≥3	X	≥3	X						
	al06	X	X	X	X				≥3								≥3	≥3			≥3	≥3									≥3	X	≥3	X						
	al07	X	X	X	X				≥3																						≥3	X	≥3	X						
	al08			≥3	X																											≥4		≥4						
	al14	X	X	X	X		≥6	≥3	≥3						≥7	≥3	≥3	≥3	≥7	≥3	≥3	≥3									≥3	X	≥3	X						
	al15		X	X	X				≥3									≥3				≥3									≥4	X	≥4	X						
	al16	X	X	X	X				≥3								≥3	≥3			≥3	≥3									≥3	X	≥3	X						
	al17	X	X	X	X		≥6	≥3	≥3						≥7	≥3	≥3	≥3	≥7	≥3	≥3	≥3									≥3	X	≥3	X						
	al18	X	X	X	X	≥6	≥6	≥3	≥3						≥7	≥3	≥3	≥3	≥7	≥3	≥3	≥3									≥3	X	≥3	X						
	al19	X	X	X	X	≥6	≥6	≥3	≥3						≥7	≥3	≥3	≥3	≥7	≥3	≥3	≥3									≥3	X	≥3	X						
AAC main	am00	X	X	X	X																										≥3	X	≥3	X						
	am01	X	X	X	X																										≥3	X	≥3	X						
	am02	X	X	X	X																										≥3	X	≥3	X						
	am03		≥1	X	X																										≥7	X	≥7	X						
	am04	X	X	X	X																										≥3	X	≥3	X						
	am05		X	X	X																										≥6	X	≥6	X						
	am06	X	X	X	X																										≥3	X	≥3	X						
	am07		X	X	X																										≥5	X	≥5	X						
AAC SSR	as00	X	X	X	X																										≥3	X	≥3	X						
	as01	X	X	X	X																										≥3	X	≥3	X						
	as02	X	X	X	X																										≥3	X	≥3	X						
	as03	X	X	X	X																										≥3	X	≥3	X						
	as04	X	X	X	X																										≥3	X	≥3	X						
	as05	X	X	X	X																										≥3	X	≥3	X						
	as06	X	X	X	X																										≥3	X	≥3	X						
	as07	X	X	X	X																										≥3	X	≥3	X						
	as08	X	X	X	X																										≥3	X	≥3	X						
	as09	X	X	X	X																										≥3	X	≥3	X						
	as10	X	X	X	X																										≥3	X	≥3	X						
	as11	X	X	X	X																										≥3	X	≥3	X						
	as12	X	X	X	X																										≥3	X	≥3	X						
	as13	≥2	X	X	X																										≥5	X	≥5	X						
	as14	≥2	X	X	X																										≥5	X	≥5	X						
	as15	≥3	X	X	X																										≥6	X	≥6	X						
	as16	≥3	X	X	X																										≥6	X	≥6	X						
	as17	X	X	X	X																										≥3	X	≥3	X						
AAC LTP	ap01	X	X	X	X			X	X							X	X	X		X	X	X									X	X	X	X						
	ap02	X	X	X	X			X	X							X	X	X		X	X	X									X	X	X	X						
	ap03	X	X	X	X			X	X							X	X	X		X	X	X									X	X	X	X						
	ap04	X	X	X	X			X	X							X	X	X		X	X	X									X	X	X	X						
	ap05	X	X	X	X			X	X							X	X	X		X	X	X									X	X	X	X						
AAC scalable	ac01	X	X	X	X			X	X							X	X	X		X	X	X									X	X	X	X						
	ac02	X	X	X	X			X	X							X	X	X		X	X	X									X	X	X	X						
	ac03	X	X	X	X			X	X							X	X	X		X	X	X									X	X	X	X						
	ac04 Layer 0	X	X	X	X	X	X	X	X						X	X	X	X	X	X	X	X									X	X	X	X						

	ac04 Layer 1 – 7	X	X	X	X			X	X						X	X	X		X	X	X								X	X	X	X							
	ac05 Layer 0	X	X	X	X	X	X	X	X					X	X	X	X	X	X	X	X								X	X	X	X							
	ac05 Layer 1 – 7	X	X	X	X			X	X						X	X	X		X	X	X								X	X	X	X							
	ac06 Layer 0	X	X	X	X	X	X	X	X					X	X	X	X	X	X	X	X								X	X	X	X							
	ac06 Layer 1 – 4	X	X	X	X		X	X	X					X	X	X	X	X	X	X	X								X	X	X	X							
	ac08	X	X	X	X			X	X						X	X	X		X	X	X								X	X	X	X							
	ac09	X	X	X	X			X	X						X	X	X		X	X	X								X	X	X	X							
	ac10 Layer 0	X	X	X	X	X	X	X	X					X	X	X	X	X	X	X	X								X	X	X	X							
	ac10 Layer 1 – 4	X	X	X	X		X	X	X					X	X	X	X	X	X	X	X								X	X	X	X							
	ac12	X	X	X	X	X	X	X	X					X	X	X	X	X	X	X	X								X	X	X	X							
	ac13 Layer 0	X	X	X	X	X	X	X	X					X	X	X	X	X	X	X	X								X	X	X	X							
	ac13 Layer 1	X	X	X	X	X	X	X	X						X	X	X		X	X	X								X	X	X	X							
	ac14 Layer 0	X	X	X	X	X	X	X	X					X	X	X	X	X	X	X	X								X	X	X	X							
	ac14 Layer 1 – 2	X	X	X	X			X	X						X	X	X		X	X	X								X	X	X	X							
	ac15 Layer 0	X	X	X	X	X	X	X	X					X	X	X	X	X	X	X	X								X	X	X	X							
	ac15 Layer 1 – 2	X	X	X	X																									X	X	X							
	ac16 Layer 0 – 1	X	X	X	X	X	X	X	X					X	X	X	X												X	X	X	X							
	ac16 Layer 2	X	X	X	X		X	X	X					X	X	X	X												X	X	X	X							
	ac17 Layer 0 – 1	X	X	X	X	X	X	X	X						X	X	X												X	X	X	X							
	ac17 Layer 2	X	X	X	X		X	X	X						X	X	X												X	X	X	X							
	ac18	X	X	X	X			X	X						X	X	X												X	X	X	X							
	ac19	X	X	X	X																								X	X	X	X							
	ac20	X	X	X	X			X	X						X	X	X												X	X	X	X							
	ac22	X	X	X	X			X	X						X	X	X												X	X	X	X							
Twin VQ	tv00	X	X	X	X	X	x	x	X																				X	X	X	X							
	tv01	X	X	X	X	X	x	x	X																				X	X	X	X							
	tv02	X	X	X	X	X	x	x	X																				X	X	X	X							
	tv03	X	X	X	X	X	x	x	X																				X	X	X	X							
	tv04	X	X	X	X	X	x	x	X																				X	X	X	X							
	tv05	X	X	X	X	X	x	x	X																				X	X	X	X							
	tv06	X	X	X	X	X	x	x	X																				X	X	X	X							
	tv07	X	X	X	X	X	x	x	X																				X	X	X	X							
	tv11	X	X	X	X		X	x	X																				X	X	X	X							
	tv14	X	X	X	X	X	x	x	X																				X	X	X	X							
	tv15	X	X	X	X			X	X																				X	X	X	X							
	tv16	X	X	X	X		X	x	X																				X	X	X	X							
	tv20	X	X	X	X	X	x	x	X																				X	X	X	X							
	tv21	X	X	X	X		X	x	X																				X	X	X	X							
	tv22	X	X	X	X			X	X																				X	X	X	X							
	tv23	X	X	X	X	X	x	x	X																				X	X	X	X							
	tv24	X	X	X	X	X	x	x	X																				X	X	X	X							
	tv25	X	X	X	X	X	x	x	X																				X	X	X	X							
	tv26	X	X	X	X			X	X																				X	X	X	X							
	tv27	X	X	X	X	X	x	x	X																				X	X	X	X							
CELP	ce00	X	X	X	X	X	X	X	X	X	X			X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X							
	ce01	X	X	X	X	X	X	X	X	X	X			X	X	X	X	X	X	X	X		X	X	X		X	X	X	X	X	X							
	ce02	X	X	X	X	X	X	X	X	X	X			X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X							
	ce03	X	X	X	X	X	X	X	X	X	X			X	X	X	X	X	X	X	X		X	X	X		X	X	X	X	X	X							
	ce04	X	X	X	X	X	X	X	X	X	X			X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X							
	ce05	X	X	X	X	X	X	X	X	X	X			X	X	X	X	X	X	X	X		X	X	X		X	X	X	X	X	X							
	ce06	X	X	X	X	X	X	X	X	X	X			X	X	X	X	X	X	X	X		X	X	X		X	X	X	X	X	X							
	ce07	X	X	X	X	X	X	X	X	X	X			X	X	X	X	X	X	X	X		X	X	X		X	X	X	X	X	X							
	ce08	X	X	X	X	X	X	X	X	X	X			X	X	X	X	X	X	X	X		X	X	X		X	X	X	X	X	X							
	ce09	X	X	X	X	X	X	X	X	X	X			X	X	X	X	X	X	X	X		X	X	X		X	X	X	X	X	X							
	ce10	X	X	X	X	X	X	X	X	X	X			X	X	X	X	X	X	X	X		X	X	X		X	X	X	X	X	X							
	ce11	X	X	X	X	X	X	X	X	X	X			X	X	X	X	X	X	X	X		X	X	X		X	X	X	X	X	X							
	ce12	X	X	X	X	X	X	X	X	X	X			X	X	X	X	X	X	X	X		X	X	X		X	X	X	X	X	X							
	ce13	X	X	X	X	X	X	X	X	X	X			X	X	X	X	X	X	X	X		X	X	X		X	X	X	X	X	X							
	ce14	X	X	X	X	X	X	X	X	X	X			X	X	X	X	X	X	X	X		X	X	X		X	X	X	X	X	X							
HVXC	hv01	X	X	X	X	X	X	X	X	X	X											X	X	X	X	X	X	X	X	X	X	X							
	hv02	X	X	X	X	X	X	X	X	X	X											X	X	X	X	X	X	X	X	X	X	X							
	hv03	X	X	X	X	X	X	X	X	X	X											X	X	X	X	X	X	X	X	X	X	X							
	hv04	X	X	X	X	X	X	X	X	X	X											X	X	X	X	X	X	X	X	X	X	X							
	hv05	X	X	X	X	X	X	X	X	X	X											X	X	X	X	X	X	X	X	X	X	X							

	hv06	X	X	X	X	X	X	X	X	X	X										X	X	X	X	X	X	X	X	X	X	X	X								
	hv07	X	X	X	X	X	X	X	X	X	X										X	X	X	X	X	X	X	X	X	X	X	X								
	hv08	X	X	X	X	X	X	X	X	X	X										X	X	X	X	X	X	X	X	X	X	X	X								
TTSI	tts1	X	X	X	X	X	X	X	X	X	X	X	X								X	X	X	X	X	X	X	X	X	X	X	X								
	tts2	X	X	X	X	X	X	X	X	X	X	X	X								X	X	X	X	X	X	X	X	X	X	X	X								
	tts3	X	X	X	X	X	X	X	X	X	X	X	X								X	X	X	X	X	X	X	X	X	X	X	X								
	tts4	X	X	X	X	X	X	X	X	X	X	X	X								X	X	X	X	X	X	X	X	X	X	X	X								
	tts5	X	X	X	X	X	X	X	X	X	X	X	X								X	X	X	X	X	X	X	X	X	X	X	X								
	tts6	X	X	X	X	X	X	X	X	X	X	X	X								X	X	X	X	X	X	X	X	X	X	X	X								
	tts7	X	X	X	X	X	X	X	X	X	X	X	X								X	X	X	X	X	X	X	X	X	X	X	X								
	tts8	X	X	X	X	X	X	X	X	X	X	X	X								X	X	X	X	X	X	X	X	X	X	X	X								
Algorithmic Synthesis	sy001	X	X	X	X						X	X	X																												
	sy002	X	X	X	X						X	X	X																												
	sy003	X	X	X	X						X	X	X																												
	sy004	X	X	X	X						X	X	X																												
	sy005			X	X							X	X																												
	sy006				X								X																												
	sy007	X	X	X	X						X	X	X																												
	sy008			X	X							X	X																												
	sy009				X								X																												
	sy010	X	X	X	X						X	X	X																												
	sy011	X	X	X	X						X	X	X																												
	sy012	X	X	X	X						X	X	X																												
	sy013	X	X	X	X						X	X	X																												
ER AAC LC	er_al10_ep0													≥7	≥3	≥3	≥3	≥7	≥3	≥3	≥3									≥3	X	≥3	X								
	er_al10_ep1													≥7	≥3	≥3	≥3	≥7	≥3	≥3	≥3									≥3	X	≥3	X								
	er_al10_ep3_crc01_08													X	X	X	X	X	X	X	X									X	X	X	X								
	er_al10_ep3_crc01_22													X	X	X	X	X	X	X	X									X	X	X	X								
	er_al12_ep0													≥7	≥3	≥3	≥3	≥7	≥3	≥3	≥3									≥3	X	≥3	X								
	er_al12_ep1													≥7	≥3	≥3	≥3	≥7	≥3	≥3	≥3									≥3	X	≥3	X								
	er_al15_ep0													≥7	≥3	≥3	≥3	≥7	≥3	≥3	≥3									≥3	X	≥3	X								
	er_al15_ep1													≥7	≥3	≥3	≥3	≥7	≥3	≥3	≥3									≥3	X	≥3	X								
	er_al15_ep3_srcpc01_08																	X	X	X	X											X	X								
	er_al15_ep3_srcpc01_48																		X	X	X											X	X								
	er_al18_ep0													≥7	≥3	≥3	≥3	≥7	≥3	≥3	≥3									≥3	X	≥3	X								
	er_al18_ep1													≥7	≥3	≥3	≥3	≥7	≥3	≥3	≥3									≥3	X	≥3	X								
	er_al18_ep3_crc02_11													X	X	X	X	X	X	X	X									X	X	X	X								
	er_al18_ep3_crc02_24														X	X	X		X	X	X									X	X	X	X								
	er_al18_ep3_srcpc02_11																	X	X	X	X											X	X								
	er_al18_ep3_srcpc02_64																																X								
	er_al21_ep0													≥7	≥3	≥3	≥3	≥7	≥3	≥3	≥3									≥3	X	≥3	X								
	er_al21_ep1													≥7	≥3	≥3	≥3	≥7	≥3	≥3	≥3									≥3	X	≥3	X								
	er_al21_ep3_crc03_12													X	X	X	X	X	X	X	X									X	X	X	X								
	er_al21_ep3_crc03_32														X	X	X		X	X	X									X	X	X	X								
	er_al23_ep0													≥7	≥3	≥3	≥3	≥7	≥3	≥3	≥3									≥3	X	≥3	X								
	er_al23_ep1													≥7	≥3	≥3	≥3	≥7	≥3	≥3	≥3									≥3	X	≥3	X								
	er_al23_ep3_crc04_16													X	X	X	X	X	X	X	X									X	X	X	X								
	er_al23_ep3_crc04_44														X	X	X		X	X	X									X	X	X	X								
	er_al26_ep0													≥7	≥3	≥3	≥3	≥7	≥3	≥3	≥3									≥3	X	≥3	X								
	er_al26_ep1													≥7	≥3	≥3	≥3	≥7	≥3	≥3	≥3									≥3	X	≥3	X								
	er_al26_ep3_srcpc03_12																	X	X	X	X											X	X								
	er_al26_ep3_srcpc03_88																																X								
ER AAC LTP	er_ap10_ep0													≥7	≥3	≥3	≥3	≥7	≥3	≥3	≥3									≥3	X	≥3	X	≥6	≥3	≥3	≥6	≥3	≥3
	er_ap10_ep1													≥7	≥3	≥3	≥3	≥7	≥3	≥3	≥3									≥3	X	≥3	X	≥6	≥3	≥3	≥6	≥3	≥3
	er_ap14_ep0													≥7	≥3	≥3	≥3	≥7	≥3	≥3	≥3									≥3	X	≥3	X	≥6	≥3	≥3	≥6	≥3	≥3
	er_ap14_ep1													≥7	≥3	≥3	≥3	≥7	≥3	≥3	≥3									≥3	X	≥3	X	≥6	≥3	≥3	≥6	≥3	≥3
	er_ap14_ep3_srs02_12																	X	X	X	X											X	X				X	X	X

	er_ap14_ep3_srs02_32																			X	X	X											X	X					X	X
	er_ap27_ep0														≥7	≥3	≥3	≥3	≥7	≥3	≥3	≥3									≥3	X	≥3	X		≥3	≥3		≥3	≥3
	er_ap27_ep1														≥7	≥3	≥3	≥3	≥7	≥3	≥3	≥3									≥3	X	≥3	X		≥3	≥3		≥3	≥3
	er_ap27_ep3_srcpc04_16																		X	X	X	X											X	X					X	X
	er_ap27_ep3_srcpc04_96																																	X						
ER AAC LD	er_ad100_ep0																								≥3	≥3			≥3	≥3	≥3	X	≥3	X	≥6	≥3	≥3	≥6	≥3	≥3
	er_ad100_ep1																								≥3	≥3			≥3	≥3	≥3	X	≥3	X	≥6	≥3	≥3	≥6	≥3	≥3
	er_ad103_ep0																								≥3	≥3			≥3	≥3	≥3	X	≥3	X	≥6	≥3	≥3	≥6	≥3	≥3
	er_ad103_ep1																								≥3	≥3			≥3	≥3	≥3	X	≥3	X	≥6	≥3	≥3	≥6	≥3	≥3
	er_ad107_ep0																								≥3	≥3			≥3	≥3	≥3	X	≥3	X	≥6	≥3	≥3	≥6	≥3	≥3
	er_ad107_ep1																								≥3	≥3			≥3	≥3	≥3	X	≥3	X	≥6	≥3	≥3	≥6	≥3	≥3
	er_ad109_ep0																								≥3	≥3			≥3	≥3	≥3	X	≥3	X	≥6	≥3	≥3	≥6	≥3	≥3
	er_ad109_ep1																								≥3	≥3			≥3	≥3	≥3	X	≥3	X	≥6	≥3	≥3	≥6	≥3	≥3
	er_ad110_ep0																								≥3	≥3			≥3	≥3	≥3	X	≥3	X	≥6	≥3	≥3	≥6	≥3	≥3
	er_ad110_ep1																								≥3	≥3			≥3	≥3	≥3	X	≥3	X	≥6	≥3	≥3	≥6	≥3	≥3
	er_ad111_ep0																								≥3	≥3			≥3	≥3	≥3	X	≥3	X	≥6	≥3	≥3	≥6	≥3	≥3
	er_ad111_ep1																								≥3	≥3			≥3	≥3	≥3	X	≥3	X	≥6	≥3	≥3	≥6	≥3	≥3
	er_ad115_ep0																								≥3	≥3			≥3	≥3	≥3	X	≥3	X	≥6	≥3	≥3	≥6	≥3	≥3
	er_ad115_ep1																								≥3	≥3			≥3	≥3	≥3	X	≥3	X	≥6	≥3	≥3	≥6	≥3	≥3
	er_ad115_ep3_srs01_16																												X	X			X	X				X	X	X
	er_ad115_ep3_srs01_44																												X	X			X	X					X	X
	er_ad202_ep0																									≥3				≥3	≥3	X	≥3	X		≥3	≥3		≥3	≥3
	er_ad202_ep1																									≥3				≥3	≥3	X	≥3	X		≥3	≥3		≥3	≥3
	er_ad206_ep0																									≥3				≥3	≥3	X	≥3	X		≥3	≥3		≥3	≥3
	er_ad206_ep1																									≥3				≥3	≥3	X	≥3	X		≥3	≥3		≥3	≥3
	er_ad206_ep3_srs03_08																													X			X	X					X	X
	er_ad206_ep3_srs03_48																													X			X	X					X	X
	er_ad214_ep0																									≥3				≥3	≥3	X	≥3	X		≥3	≥3		≥3	≥3
	er_ad214_ep1																									≥3				≥3	≥3	X	≥3	X		≥3	≥3		≥3	≥3
	er_ad218_ep0																									≥3				≥3	≥3	X	≥3	X		≥3	≥3		≥3	≥3
	er_ad218_ep1																									≥3				≥3	≥3	X	≥3	X		≥3	≥3		≥3	≥3
	er_ad218_ep3_srs04_11																													X			X	X					X	X
	er_ad218_ep3_srs04_64																																	X						
ER AAC scalable	er_ac111_ep0															X	X	X		X	X	X									X	X	X	X		X	X		X	X
	er_ac111_ep1															X	X	X		X	X	X									X	X	X	X		X	X		X	X
	er_ac111_ep3_cplx01																			X	X	X											X	X					X	X
	er_ac118_ep0																															X		X						
	er_ac118_ep1																															X		X						
	er_ac119_ep0 Layer 0-1														X	X	X	X	X	X	X	X									X	X	X	X	X	X	X	X	X	X
	er_ac119_ep0 Layer 2-3														X	X	X	X	X	X	X	X									X	X	X	X		X	X		X	X
	er_ac119_ep1 Layer0-1														X	X	X	X	X	X	X	X									X	X	X	X	X	X	X	X	X	X
	er_ac119_ep1 Layer 2-3														X	X	X	X	X	X	X	X									X	X	X	X		X	X		X	X
	er_ac121_ep0															X	X	X		X	X	X									X	X	X	X		X	X		X	X
	er_ac123_ep0															X	X	X		X	X	X									X	X	X	X		X	X		X	X
	er_ac123_ep1															X	X	X		X	X	X									X	X	X	X		X	X		X	X
	er_ac123_ep3_cplx02																			X	X	X											X	X					X	X
	er_ac211_ep0 Layer 0														X	X	X	X	X	X	X	X									X	X	X	X	X	X	X	X	X	X
	er_ac211_ep0 Layer 1															X	X	X		X	X	X									X	X	X	X	X	X	X	X	X	X
	er_ac211_ep0 Layer 2															X	X	X		X	X	X									X	X	X	X		X	X		X	X
	er_ac211_ep1 Layer 0														X	X	X	X	X	X	X	X									X	X	X	X	X	X	X	X	X	X
	er_ac211_ep1 Layer 1															X	X	X		X	X	X									X	X	X	X	X	X	X	X	X	X
	er_ac211_ep1 Layer 1															X	X	X		X	X	X									X	X	X	X		X	X		X	X
	er_ac211_ep3_cplx03 Layer 0														X	X	X	X	X	X	X	X									X	X	X	X	X	X	X	X	X	X

	er_ac211_ep3_cplx03 Layer 1																			X	X	X											X	X				X	X	X
	er_ac211_ep3_cplx03 Layer 2															X	X	X		X	X	X									X	X	X	X		X	X		X	X
	er_ac221_ep0 Layer 0														X	X	X	X	X	X	X	X									X	X	X	X	X	X	X	X	X	X
	er_ac221_ep0 Layer 1-2														X	X	X	X	X	X	X	X									X	X	X	X		X	X		X	X
	er_ac221_ep1 Layer 0														X	X	X	X	X	X	X	X									X	X	X	X	X	X	X	X	X	X
	er_ac221_ep1 Layer 1-2														X	X	X	X	X	X	X	X									X	X	X	X		X	X		X	X
	er_ac221_ep3_cplx04 Layer 0														X	X	X	X	X	X	X	X									X	X	X	X	X	X	X	X	X	X
	er_ac221_ep3_cplx04 Layer 1																		X	X	X	X										X		X					X	X
	er_ac221_ep3_cplx04 Layer 2														X	X	X	X	X	X	X	X									X	X	X	X		X	X		X	X
	er_ac311_ep0															X	X	X		X	X	X									X	X	X	X		X	X		X	X
	er_ac311_ep1															X	X	X		X	X	X									X	X	X	X		X	X		X	X
	er_ac311_ep3_cplx05 Layer 0															X	X	X		X	X	X									X	X	X	X		X	X		X	X
	er_ac311_ep3_cplx05 Layer 1-2																			X	X	X											X	X					X	X
	er_ac321_ep0 Layer 0															X	X	X		X	X	X									X	X	X	X	X	X	X	X	X	X
	er_ac321_ep0 Layer 1-2															X	X	X		X	X	X									X	X	X	X		X	X		X	X
	er_ac321_ep1 Layer 0															X	X	X		X	X	X									X	X	X	X	X	X	X	X	X	X
	er_ac321_ep1 Layer 1-2															X	X	X		X	X	X									X	X	X	X		X	X		X	X
	er_ac321_ep3_cplx06 Layer 0															X	X	X		X	X	X									X	X	X	X		X	X		X	X
	er_ac321_ep3_cplx06 Layer 1-2																			X	X	X											X	X					X	X
ER TwinVQ	er_tv01_ep0																														X	X	X	X	X	X	X	X	X	X
	er_tv01_ep1																														X	X	X	X	X	X	X	X	X	X
	er_tv01_ep2																																X	X				X	X	X
	er_tv02_ep0																														X	X	X	X	X	X	X	X	X	X
	er_tv02_ep1																														X	X	X	X	X	X	X	X	X	X
	er_tv02_ep2																																x	X				X	X	X
ER BSAC	er_bs01_ep0																														X	X			≥6	<6	<6			
	er_bs01_ep1																														X	X			≥6	<6	<6			
	er_bs01_ep2																																X	X				≥6	<6	<6
	er_bs02_ep0																														X	X			≥6	<6	<6			
	er_bs02_ep1																														X	X			≥6	<6	<6			
	er_bs02_ep2																																X	X				≥6	<6	<6
	er_bs03_ep0																														X	X				X	X			
	er_bs03_ep1																														X	X				X	X			
	er_bs03_ep2																																X	X					X	X
	er_bs04_ep0																														X	X				X	X			
	er_bs04_ep1																														X	X				X	X			
	er_bs04_ep2																																X	X					X	X
	er_bs05_ep0																														X	X				X	X			
	er_bs05_ep1																														X	X				X	X			
	er_bs05_ep2																																X	X					X	X
	er_bs06_ep0																														X	X				X	X			
	er_bs06_ep1																														X	X				X	X			
	er_bs06_ep2																																X	X					X	X
ER CELP	er_ce00_ep0														X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X						
	er_ce00_ep1														X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X						
	er_ce00_ep2																		X	X	X	X					X	X	X	X			X	X						
	er_ce01_ep0														X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X						
	er_ce01_ep1														X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X						
	er_ce02_ep0														X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X						
	er_ce02_ep1														X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X						
	er_ce02_ep2																		X	X	X	X					X	X	X	X			X	X						
	er_ce03_ep0														X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X						
	er_ce03_ep1														X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X						

	er_ce04_ep0														X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X						
	er_ce04_ep1														X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X						
	er_ce05_ep0														X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X						
	er_ce05_ep1														X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X						
	er_ce06_ep0														X	X	X	X	X	X	X	X		X	X	X		X	X	X	X	X	X	X						
	er_ce06_ep1														X	X	X	X	X	X	X	X		X	X	X		X	X	X	X	X	X	X						
	er_ce06_ep2																		X	X	X	X						X	X	X			X	X						
	er_ce07_ep0														X	X	X	X	X	X	X	X		X	X	X		X	X	X	X	X	X	X						
	er_ce07_ep1														X	X	X	X	X	X	X	X		X	X	X		X	X	X	X	X	X	X						
	er_ce08_ep0														X	X	X	X	X	X	X	X		X	X	X		X	X	X	X	X	X	X						
	er_ce08_ep1														X	X	X	X	X	X	X	X		X	X	X		X	X	X	X	X	X	X						
	er_ce09_ep0														X	X	X	X	X	X	X	X		X	X	X		X	X	X	X	X	X	X						
	er_ce09_ep1														X	X	X	X	X	X	X	X		X	X	X		X	X	X	X	X	X	X						
	er_ce09_ep2																		X	X	X	X						X	X	X			X	X						
	er_ce10_ep0														X	X	X	X	X	X	X	X		X	X	X		X	X	X	X	X	X	X						
	er_ce10_ep1														X	X	X	X	X	X	X	X		X	X	X		X	X	X	X	X	X	X						
	er_ce11_ep0														X	X	X	X	X	X	X	X		X	X	X		X	X	X	X	X	X	X						
	er_ce11_ep1														X	X	X	X	X	X	X	X		X	X	X		X	X	X	X	X	X	X						
	er_ce12_ep0														X	X	X	X	X	X	X	X		X	X	X		X	X	X	X	X	X	X						
	er_ce12_ep1														X	X	X	X	X	X	X	X		X	X	X		X	X	X	X	X	X	X						
	er_ce13_ep0														X	X	X	X	X	X	X	X		X	X	X		X	X	X	X	X	X	X						
	er_ce13_ep1														X	X	X	X	X	X	X	X		X	X	X		X	X	X	X	X	X	X						
	er_ce13_ep2																		X	X	X	X						X	X	X			X	X						
	er_ce14_ep0														X	X	X	X	X	X	X	X		X	X	X		X	X	X	X	X	X	X						
	er_ce14_ep1														X	X	X	X	X	X	X	X		X	X	X		X	X	X	X	X	X	X						
	er_ce15_ep0														X	X	X	X	X	X	X	X		X	X	X		X	X	X	X	X	X	X						
	er_ce15_ep1														X	X	X	X	X	X	X	X		X	X	X		X	X	X	X	X	X	X						
	er_ce15_ep2																		X	X	X	X						X	X	X			X	X						
	er_ce16_ep0														X	X	X	X	X	X	X	X		X	X	X		X	X	X	X	X	X	X						
	er_ce16_ep1														X	X	X	X	X	X	X	X		X	X	X		X	X	X	X	X	X	X						
	er_ce17_ep0														X	X	X	X	X	X	X	X		X	X	X		X	X	X	X	X	X	X						
	er_ce17_ep1														X	X	X	X	X	X	X	X		X	X	X		X	X	X	X	X	X	X						
	er_ce18_ep0														X	X	X	X	X	X	X	X		X	X	X		X	X	X	X	X	X	X						
	er_ce18_ep1														X	X	X	X	X	X	X	X		X	X	X		X	X	X	X	X	X	X						
	er_ce18_ep2																		X	X	X	X						X	X	X			X	X						
ER HVXC	er_hv01_ep0																						X	X	X	X	X	X	X	X	X	X	X	X						
	er_hv01_ep1																						X	X	X	X	X	X	X	X	X	X	X	X						
	er_hv02_ep0																						X	X	X	X	X	X	X	X	X	X	X	X						
	er_hv02_ep1																						X	X	X	X	X	X	X	X	X	X	X	X						
	er_hv03_ep0																						X	X	X	X	X	X	X	X	X	X	X	X						
	er_hv03_ep1																						X	X	X	X	X	X	X	X	X	X	X	X						
	er_hv03_ep2_crc																										X	X	X	X			X	X						
	er_hv03_ep2_fec																										X	X	X	X			X	X						
	er_hv04_ep0																						X	X	X	X	X	X	X	X	X	X	X	X						
	er_hv04_ep1																						X	X	X	X	X	X	X	X	X	X	X	X						
	er_hv04_ep2_crc																										X	X	X	X			X	X						
	er_hv04_ep2_fec																										X	X	X	X			X	X						
	er_hv05_ep0																						X	X	X	X	X	X	X	X	X	X	X	X						
	er_hv05_ep1																						X	X	X	X	X	X	X	X	X	X	X	X						
	er_hv06_ep0																						X	X	X	X	X	X	X	X	X	X	X	X						
	er_hv06_ep1																						X	X	X	X	X	X	X	X	X	X	X	X						
	er_hv07_ep0																						X	X	X	X	X	X	X	X	X	X	X	X						
	er_hv07_ep1																						X	X	X	X	X	X	X	X	X	X	X	X						
	er_hv08_ep0																						X	X	X	X	X	X	X	X	X	X	X	X						
	er_hv08_ep1																						X	X	X	X	X	X	X	X	X	X	X	X						
	er_hv09_ep0																						X	X	X	X	X	X	X	X	X	X	X	X						
	er_hv09_ep1																						X	X	X	X	X	X	X	X	X	X	X	X						
	er_hv10_ep0																						X	X	X	X	X	X	X	X	X	X	X	X						
	er_hv10_ep1																						X	X	X	X	X	X	X	X	X	X	X	X						
ER HILN	er_hi00_ep0																														X	X	X	X						
	er_hi01_ep0																														X	X	X	X						
	er_hi02_ep0																														X	X	X	X						
	er_hi03_ep0																														X	X	X	X						
	er_hi04_ep0																														X	X	X	X						

	er_hi05_ep0																													X	X	X	X						
	er_hi06_ep0																													X	X	X	X						
	er_hi07_ep0																													X	X	X	X						
	er_hi10_ep0																													X	X	X	X						
	er_hi11_ep0																													X	X	X	X						
	er_hi12_ep0																													X	X	X	X						
	er_hi13_ep0																													X	X	X	X						
	er_hi14_ep0																													X	X	X	X						
	er_hi15_ep0																													X	X	X	X						
	er_hi16_ep0																													X	X	X	X						
	er_hi20_ep0																													X	X	X	X						
	er_hi21_ep0																													X	X	X	X						
	er_hi22_ep0																													X	X	X	X						
	er_hi23_ep0																													X	X	X	X						
	er_hi24_ep0																													X	X	X	X						
	er_hi25_ep0																													X	X	X	X						
	er_hi26_ep0																													X	X	X	X						
	er_hi26_ep1																													X	X	X	X						
	er_hi26_ep3_crc																															X	X						
	er_hi26_ep3_fec																															X	X						
	er_hi27_ep0																													X	X	X	X						
	er_hi27_ep1																													X	X	X	X						
	er_hi27_ep3_crc																															X	X						
	er_hi27_ep3_fec																															X	X						
	er_hi28_ep0																													X	X	X	X						
	er_hi28_ep1																													X	X	X	X						
	er_hi28_ep3_crc																															X	X						
	er_hi28_ep3_fec																															X	X						
	er_hi29_ep0																													X	X	X	X						
	er_hi29_ep1																													X	X	X	X						
	er_hi29_ep3_crc																															X	X						
	er_hi29_ep3_fec																															X	X						
	er_hi30_ep0																														X		X						
	er_hi31_ep0																														X		X						
ER Parametric	er_pa00_ep0																													X	X	X	X						
	er_pa01_ep0																													X	X	X	X						
	er_pa02_ep0																													X	X	X	X						
	er_pa03_ep0																													X	X	X	X						

6.12.2 Systems

Table 105

Object type	file name	Main Scene Graph Profile				Audio Scene Graph Profile				3-D Audio Scene Graph Profile			
Level		**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**
AudioSource and Sound2D	ab001	x	x	x	x	x	x	x	x				
	ab002	x	x	x	x	x	x	x	x				
	ab003	x	x	x	x	x	x	x	x				
	ab004	x	x	x	x	x	x	x	x				
	ab005		x	x	x		x	x	x				
	ab006		x	x	x		x	x	x				
AudioSource and Sound	ab011	x	x	x	x								
	ab012	x	x	x	x								
	ab013	x	x	x	x								
	ab014	x	x	x	x								
	ab015		x	x	x								
	ab016		x	x	x								
Audio Switch	ab031		x	x	x		x	x	x	x	x	x	x
Audio Mix and SR conversion	ab041		x	x	x		x	x	x	x	x	x	x
	ab042		x	x	x		x	x	x	x	x	x	x
	ab043		x	x	x		x	x	x	x	x	x	x
	ab044		x	x	x		x	x	x	x	x	x	x
Audio FX	ab101		x	x	x		x	x	x	x	x	x	x
	ab102		x	x	x		x	x	x	x	x	x	x
DirectiveSound	AABPhy1									x	x	x	x
	AABPhy2									x	x	x	x
	AABPhy3									x	x	x	x
	AABPhy4									x	x	x	x
	AABPhy5									x	x	x	x
	AABPhy6									x	x	x	x
	AABPhy7									x	x	x	x
	AABPhy8									x	x	x	x
	AABPhy9									x	x	x	x
	AABPhy10									x	x	x	x
	AABPhy11									x	x	x	x
	AABPhy12									x	x	x	x
	AABPhy13									x	x	x	x
	AABPhy14									x	x	x	x
	AABPhy15									x	x	x	x
	AABPhy16									x	x	x	x
	AABPhy17									x	x	x	x
	AABPhy18									x	x	x	x
	AABPhy19									x	x	x	x
	AABPhy20									x	x	x	x
	AABPhy21									x	x	x	x
	AABPhy22									x	x	x	x
	AABPhy23									x	x	x	x

	AABPhy24								x	x	x	x
	AABPhy25								x	x	x	x
	AABPhy26								x	x	x	x
	AABPhy27								x	x	x	x
	AABPhy28								x	x	x	x
	AABPhy29								x	x	x	x
	AABPhy30								x	x	x	x
	AABPhy31								x	x	x	x
	AABPhy32								x	x	x	x
	AABPhy33								x	x	x	x
	AABPhy34								x	x	x	x
	AABPhy35								x	x	x	x
	AABPhy36								x	x	x	x
	AABPhy37								x	x	x	x
	AABPhy38								x	x	x	x
	AABPhy39								x	x	x	x
	AABPhy40								x	x	x	x
	AABPhy41								x	x	x	x
	AABPhy42								x	x	x	x
	AABPhy43								x	x	x	x
	AABPhy44								x	x	x	x
	AABPhy45								x	x	x	x
	AABPhy46								x	x	x	x
	AABPhy47								x	x	x	x
	AABPhy48								x	x	x	x
	AABPhy49								x	x	x	x
	AABPhy50								x	x	x	x
	AABPhy51								x	x	x	x
	AABPhy52								x	x	x	x
	AABPhy53								x	x	x	x
	AABPhy54								x	x	x	x
AcousticScene	AABPhy55								x	x	x	x
	AABPhy56								x	x	x	x
	AABPhy57								x	x	x	x
	AABPhy58								x	x	x	x
	AABPhy59								x	x	x	x
	AABPhy60								x	x	x	x
	AABPhy61								x	x	x	x
	AABPhy62								x	x	x	x
	AABPhy63								x	x	x	x
	AABPhy64								x	x	x	x
AcousticMaterial	AABPhy65								x	x	x	x
	AABPhy66								x	x	x	x
	AABPhy67								x	x	x	x
	AABPhy68								x	x	x	x
	AABPhy69								x	x	x	x
	AABPhy70								x	x	x	x
	AABPhy71								x	x	x	x
	AABPhy72								x	x	x	x
	AABPhy73								x	x	x	x
	AABPhy74								x	x	x	x
	AABPhy75								x	x	x	x

	AABPhy76								x	x	x	x
	AABPhy77								x	x	x	x
	AABPhy78								x	x	x	x
	AABPhy79								x	x	x	x
	AABPhy80								x	x	x	x
DirectiveSound	AABPer1								x	x	x	x
	AABPer2								x	x	x	x
	AABPer3								x	x	x	x
	AABPer4								x	x	x	x
	AABPer5								x	x	x	x
	AABPer6								x	x	x	x
	AABPer7								x	x	x	x
	AABPer8								x	x	x	x
	AABPer9								x	x	x	x
	AABPer10								x	x	x	x
	AABPer11								x	x	x	x
	AABPer12								x	x	x	x
	AABPer13								x	x	x	x
	AABPer14								x	x	x	x
	AABPer15								x	x	x	x
	AABPer16								x	x	x	x
	AABPer17								x	x	x	x
	AABPer18								x	x	x	x
	AABPer19								x	x	x	x
	AABPer20								x	x	x	x
	AABPer21								x	x	x	x
	AABPer22								x	x	x	x
	AABPer23								x	x	x	x
	AABPer24								x	x	x	x
	AABPer25								x	x	x	x
	AABPer26								x	x	x	x
	AABPer27								x	x	x	x
	AABPer28								x	x	x	x
	AABPer29								x	x	x	x
	AABPer30								x	x	x	x
	AABPer31								x	x	x	x
	AABPer32								x	x	x	x
	AABPer33								x	x	x	x
	AABPer34								x	x	x	x
	AABPer35								x	x	x	x
	AABPer36								x	x	x	x
	AABPer37								x	x	x	x
	AABPer38								x	x	x	x
	AABPer39								x	x	x	x
	AABPer40								x	x	x	x
	AABPer41								x	x	x	x
	AABPer42								x	x	x	x
	AABPer43								x	x	x	x
	AABPer44								x	x	x	x
	AABPer45								x	x	x	x
	AABPer46								x	x	x	x
PerceptualParameters	AABPer47								x	x	x	x

	AABPer48								x	x	x	x
	AABPer49								x	x	x	x
	AABPer50								x	x	x	x
	AABPer51								x	x	x	x
	AABPer52								x	x	x	x
	AABPer53								x	x	x	x
	AABPer54								x	x	x	x
	AABPer55								x	x	x	x
	AABPer56								x	x	x	x
	AABPer57								x	x	x	x
	AABPer58								x	x	x	x
	AABPer59								x	x	x	x
	AABPer60								x	x	x	x
	AABPer61								x	x	x	x
	AABPer62								x	x	x	x
	AABPer63								x	x	x	x
	AABPer64								x	x	x	x
	AABPer65								x	x	x	x
	AABPer66								x	x	x	x
	AABPer67								x	x	x	x
	AABPer68								x	x	x	x
	AABPer69								x	x	x	x
	AABPer70								x	x	x	x
	AABPer71								x	x	x	x
	AABPer72								x	x	x	x
	AABPer73								x	x	x	x
	AABPer74								x	x	x	x
	AABPer75								x	x	x	x
	AABPer76								x	x	x	x

7 DMIF

7.1 Introduction

This clause of ISO/IEC 14496-4 defines compliance to Delivery Multimedia Integration Framework (DMIF) part 6 standard in 2 steps: the static review and the dynamic review of an implementation as defined in ISO/IEC 9646 Conformance Testing standard. The static review requirements are specified in subclause 7.2 in the form of Protocol Implementation Conformance Statement (PICS) proforma. The abstract test cases used for dynamic review are described in subclause 7.3.

A PICS which conforms to this specification shall be technically equivalent to the ISO published PICS proforma and shall preserve the numbering and ordering of the items in the ISO PICS proforma.

A PICS which conforms to this part of ISO/IEC 14496-4 shall:

a) Describe an implementation which conforms to ISO/IEC 14496-6,

b) Be a conforming PICS proforma, which has been implemented in accordance with the instruction for completion given in this clause.

c) Include the information necessary to uniquely identify both the supplier and the implementation.

This standard does not specify all the requirements with which terminal equipment intended for use in conjunction with multimedia information retrieval services has to comply. In particular, this standard does not specify (lower layer) protocols to be used to deliver/transport DMIF signalling protocol data units. Neither does it specify requirements related to safety, protection, and electromagnetic compatibility (EMC) of the equipment, nor regulatory requirements with which such equipment may be required to comply.

7.2 The PICS

This part of ISO/IEC 14496-6 conformance defines a Protocol Implementation Conformance Statement (PICS) proforma for the detailed expression of the conformance requirements of ISO/IEC 14496-6. The PICS proforma is in compliance with the relevant requirements, and in accordance with the relevant guidance for a PICS proforma, given in ISO 9646-2. The PICS proforma is a document in the form of a questionnaire designed for a given protocol by the protocol specifier. Prior to testing, the supplier or implementor of SUTs or IUTs should complete the PICS proforma by marking the capabilities and options which have been implemented and those that have not been implemented. CONNECTs to the PICS proforma are usually a simple YES, NO, NA (No CONNECT required), an exact value or a range of values. When a PICS proforma is completed, it becomes a PICS. The PICS is then used to evaluate the static conformance of an IUT and as a basis that the test suite specifier uses to develop the conformance abstract test suite (ATS). The PICS for each IUT can also help the test operator or the test laboratory to choose appropriate test cases to be executed.

The DMIF specification covered by the PICS proforma is a collection of Interface and protocol to accomplish the control of channels for delivering streams. DMIF can be viewed as consisting of two broad categories of functionality: DMIF Application Interface (DAI), and DMIF Signalling messages.

7.2.1 Global statement of conformance

Table 106 — DMIF

Item Number	Does the implementation support ...	Condition for status	Status	ISO/IEC 14496-6 reference	Implemented? Y=Yes, N=No, n/a=not applicable
Fu1	**DAI**	Required for uniform stream delivery in remote interactive, broadcast and file access implementations	o	10.3	
Fu2	**DS**	Required for all interactive operations only	c:m	13.1	
Fu3	**Q.2931 (DMIF Extensions)**	Required when ATM Q.2931 is used complemented with DS	c:m	13.3	
m. It is mandatory to support c:m It is mandatory to support in specific instances					

7.2.2 DMIF-Application Interface

7.2.2.1 DAI Conformance

In DMIF V1 the DAI is specified as a semantic interface, and no constraint on the syntax or language or operating system is imposed. However, any conformance testing of DAI shall account for the realisation of DAI on a specific platform, and on a specific syntax: subclause 7.2.2.4 provides the specific DAI syntax that is instrumental to the execution of the conformance tests.

The DAI is the same on all DMIF terminals whether the terminal is operating in a client or server role or it is profiled as

- Broadcast only
- File System only
- Interactive only
- any combination of the above

The DAI is not a mandatory conformance point.

7.2.2.2 Primitive support for DAI

Table 107 — DAI Primitives

Item	DAI Primitives Does the system support...	ISO/IEC 14496-6 reference	Condition for the status	Status	C++ DAI API
Pr1	DA_ServiceAttach (In)	10.4.1	DAI	m	7.2.2.4.1.1
Pr1.1	DA_ServiceAttach (Out)	10.4.1	Pr1	m	7.2.2.4.1.4
Pr2	DA_ServiceAttachCallback (In)	10.4.2	DAI Remote Int	m	7.2.2.4.1.2
Pr2.1	DA_ServiceAttachCallback (Out)	10.4.2	Pr2	m	7.2.2.4.1.3
Pr3	DA_ServiceDetach (In)	10.4.3	Pr1	m	7.2.2.4.1.5
Pr3.1	DA_ServiceDetach (Out)	10.4.3	Pr3	m	7.2.2.4.1.8
Pr4	DA_ServiceDetachCallback (In)	10.4.4	Pr2	m	7.2.2.4.1.6
Pr4.1	DA_ServiceDetachCallback (Out)	10.4.4	Pr4	m	7.2.2.4.1.7
Pr5	DA_ChannelAdd (In)	10.4.5	Pr1	m	7.2.2.4.2.1
Pr5.1	DA_ChannelAdd (Out)	10.4.5	Pr5	m	7.2.2.4.2.4
Pr6	DA_ChannelAddCallback (In)	10.4.6	Pr2	m	7.2.2.4.2.2
Pr6.1	DA_ChannelAddCallback (Out)	10.4.6	Pr6	m	7.2.2.4.2.3
Pr7	DA_ChannelDelete (In)	10.4.7	Pr1	m	7.2.2.4.2.5
Pr7.1	DA_ChannelDelete (Out)	10.4.7	Pr7	m	7.2.2.4.2.8
Pr8	DA_ChannelDeleteCallback (In)	10.4.8	Pr2	m	7.2.2.4.2.6
Pr8.1	DA_ChannelDeleteCallback (Out)	10.4.8	Pr8	m	7.2.2.4.2.7
Pr9	DA_UserCommand (In)	10.4.9	Pr5	m	7.2.2.4.3.1
Pr9.1	DA_UserCommandCallback (In)	10.4.10	Pr5	m	7.2.2.4.3.2
Pr10	DA_Data (In)	10.4.11	Pr5	m	7.2.2.4.4.1
Pr10.1	DA_DataCallback (In)	10.4.12	Pr5	m	7.2.2.4.4.2

7.2.2.3 Parameter support for DAI Message Primitives

Table 108 — DAI Parameters and equivalent C++ API to DMIF syntax

Item	Parameter	ISO/IEC 14496-6 reference	Condition for the status	Status	C++ API to DMIF Syntax
DA_ServiceAttach Group					
Par1	URL	Annex C		m	URL
Par2	uuDataInBuffer	10.3		m	uuData
Par3	uuDataInLen	10.3		m	uuDataLength
Par4	serviceSessionId	10.3		m	ServiceSessionID
Par5	uuDataOutBuffer	10.3		m	uuData
Par6	uuDataOutLen	10.3		m	uuDataLength
DA_ServiceAttachCallback Group					
Par7	serviceSessionId	10.3		m	ServiceSessionID
Par8	serviceName	10.3		m	ServiceName
Par9	uuDataInBuffer	10.3		m	uuData
Par10	uuDataInLen	10.3		m	uuDataLength
Par11	Response	10.3		m	Response
Par12	uuDataOutBuffer	10.3		m	uuData
Par13	uuDataOutLen	10.3		m	uuDataLength
DA_ServiceDetach Group					
Par14	serviceSessionId	10.3		m	ServiceSessionID
Par15	reason	10.3		m	Reason
Par16	Response	10.3		m	Response
DA_ServiceDetachCallback Group					
Par17	serviceSessionId	10.3		m	ServiceSessionID
Par18	reason	10.3		m	Reason
Par19	Response	10.3		m	Response
DA_ChannelAdd Group					
Par20	serviceSessionId	10.3		m	ServiceSessionID
Par21	channelHandle	10.3		m	ChannelHandle
Par22	qosDescriptor	10.2.5		m	QoS_Info
Par23	direction	10.3		m	Direction
Par24	uuDataInBuffer	10.3		m	uuData
Par25	uuDataInLen	10.3		m	uuDataLength
Par26	Response	10.3		m	Response
Par27	uuDataOutBuffer	10.3		m	uuData
Par28	uuDataOutLen	10.3		m	uuDataLength
DA_ChannelAddCallback Group					
Par20?	serviceSessionId	10.3		m	ServiceSessionID
Par21?	channelHandle	10.3		m	ChannelID
Par22?	qosDescriptor	10.2.5		m	QoS_Info
Par23?	direction	10.3		m	Direction
Par24?	uuDataInBuffer	10.3		m	uuData
Par25?	uuDataInLen	10.3		m	uuDataLength
Par26?	Response	10.3		m	Response
Par27?	uuDataOutBuffer	10.3		m	uuData
Par28?	uuDataOutLen	10.3		m	uuDataLength

DA_ChannelDelete Group					
Par29	channelHandle	10.3		m	ChannelID
Par30	reason	10.3		m	Reason
Par31	Response	10.3		m	Response
DA_ChannelDeleteCallback Group					
Par32	channelHandle	10.3		m	ChannelID
Par33	reason	10.3		m	Reason
Par34	Response	10.3		m	Response
DA_Data					
Par35	channelHandle	10.3		m	ChannelID
Par36	streamDataBuffer	10.3		m	Data
Par37	streamDataLength	10.3		m	DataLen
DA_DataCallback					
Par38	channelHandle	10.3		m	ChannelID
Par39	streamDataBuffer	10.3		m	Data
Par40	streamDataLegth	10.3		m	DataLen
Par51	errorFlag	10.3		m	ErrorFlag

7.2.2.4 DAI Syntax for conformance testing

A DAI syntax is specified to enable the conformance testing. The DAI primitives are specified in ANSI C++ in the following subclauses.

7.2.2.4.1 Service primitives

7.2.2.4.1.1 DAI_ServiceAttachReq

This function is called by the application to request the initialisation of a service session with the remote peer specified by the `[URL]` parameter. The user may provide additional information in the `[uuData]` field. This information is opaque to the DMIF layer and is only consumed by the peer user.

```
int DAI_ServiceAttachReq (const char* URL,
                          const BYTE* uuData, DWORD uuDataLength);
```

Parameters

[URL] A zero-terminated character string representing the service and its location. See Annex C for URL-Format. The URL itself is case-insensitive.

[uuData] Pointer to the user Data in case the user needs to transfer additional information. If no Data is sent, its value should be equal to NULL.

[uuDataLength] The length of the provided Data. If no Data is sent, its value should be equal to zero.

Return Values

If the DMIF Service accepts a request, the return value is `RESULT_OK`. Otherwise, it returns one of the following error codes.

NO_MORE_SESSIONS_SUPPORTED: This value indicates that a memory allocation failure occurred. In this case, the user should release a pre-established service session by calling the DAI_ServiceReleaseReq function and then try again.

CONNECTION_REFUSED: This value indicates that the attempt to connect to the destination address failed.

WRONG_SERVICE: Requested URL is not addressed to this service.

7.2.2.4.1.2 DAI_ServiceAttachInd

This function is called by the DMIF layer to notify the application of the service session request invoked by the peer entity via the `DAI_ServiceAttachReq` function. DMIF assigns a local unique service session identifier value and copies it into the `[ServiceSessionID]` parameter. The application should save this value and

always refer to it in subsequent interactions with the DMIF concerning this service session, like Data channel establishment.

```
void DAI_ServiceAttachInd (DWORD ServiceSessionID, const char* ServiceName,
                           const BYTE* uuData, DWORD uuDataLength);
```

Parameters

[ServiceSessionID] Identifies the service session identifier.

- [ServiceName] A zero-terminated character string representing the service name.

[uuData] If the user specified additional Data in the DAI_ServiceAttachReq function, this parameter points to them.

[uuDataLength] If the [Data] parameter contains valid Data, this parameter identifies the bytes of the received Data. Otherwise it is equal to zero.

7.2.2.4.1.3 DAI_ServiceAttachRsp

This function is called by the application as a Response to the DAI_ServiceAttachInd function. The [ServiceSessionID] value should be equal to the [ServiceSessionID] value provided by the DAI_ServiceAttachInd function. If the user decides to accept the service activation with the remote peer, the [Response] value should be equal to RESPONSE_OK, otherwise to RESPONSE_ERROR. The user may also provide additional information in the [uuData] field. This information is opaque to the DMIF layer and is only consumed by the peer user requesting the service.

```
int DAI_ServiceAttachRsp (DWORD ServiceSessionID, const BYTE* uuData,
                          DWORD uuDataLength, WORD Response);
```

Parameters

[ServiceSessionID] Identifies the service session identifier.

[Data] Points to the user Data in case the user needs to transfer additional information. If no Data is specified, its value should be equal to NULL.

[uuDataLength] The bytes of the provided Data. If no Data is defined, its value should be equal to zero.

[Response] Identifies the Response value (RESPONSE_OK or RESPONSE_ERROR).

Return Values

If no error occurs it returns RESULT_OK. Otherwise, it returns one of the following error codes.

INVALID_SESSION_ID: This value indicates that the user specified an invalid service session identifier in the [ServiceSessionID] parameter.

DMIF_INTERNAL_ERROR: This value indicates that an error, possibly related to memory allocation, occurred in the DMIF layer. In this case the user must close the application.

7.2.2.4.1.4 DAI_ServiceAttachCnf

This function is called by the DMIF layer to reply to the original DAI_ServiceAttachReq function. The [Response] parameter contains the Response code as specified by the peer user in the DAI_ServiceAttachRsp function. In case of a positive Response the service initiation phase is completed successfully. In this case DMIF assigns a locally significant unique service session identifier value and copies it in the [ServiceSessionID] parameter. The application should save this value and always refer to it in subsequent interactions with the DMIF concerning this service session, like Data channel establishment.

```
void DAI_ServiceAttachCnf (UINT ServiceSessionID, WORD Response,
                           const char* ServiceName,
                           const BYTE* uuData, DWORD uuDataLength);
```

Parameters

[ServiceSessionID] Identifies the service session identifier.

- [Response] Identifies the Response value (RESPONSE_OK or RESPONSE_ERROR).
- [ServiceName] A zero-terminated character string representing the service name.

- [uuData] If the user specified additional Data in the DAI_ServiceAttachRsp function this parameter points to them.
- [uuDataLength] If the [uuData] parameter contains valid Data, this parameter identifies the bytes of the received Data. Otherwise, it is equal to zero.

7.2.2.4.1.5 DAI_ServiceDetachReq

This function is called by the application to detach a service session identified by the [ServiceSessionID] parameter.

```
int DAI_ServiceDetachReq (DWORD ServiceSessionID, WORD Reason);
```

Parameters

[ServiceSessionID] Identifies the service session that is going to be terminated.

[Reason] Identifies the reason for the service session termination. Default value is REASON_OK.

Return Values

If no error occurs it returns RESULT_OK. Otherwise, it returns one of the following error codes.

INVALID_SESSION_ID: This value indicates that the user specified an invalid service session identifier in the [ServiceSessionID] parameter.

7.2.2.4.1.6 DAI_ServiceDetachInd

This function is called by the DMIF to inform the application that the termination of a service session identified by the [ServiceSessionID] parameter is requested by the peer entity. The [reason] parameter is the one specified by the peer user in the DAI_ServiceDetachReq function.

```
void DAI_ServiceDetachInd (DWORD ServiceSessionID, WORD Reason);
```

Parameters

[ServiceSessionID] Identifies the service session that is going to be terminated.

[Reason] Identifies the reason for the service session termination. Default value is REASON_OK.

7.2.2.4.1.7 DAI_ServiceDetachRsp

This function is called by the application as a response to the DAI_ServiceDetachRsp function. In case of a positive response, DMIF frees all resources used for this service session.

```
int DAI_ServiceDetachRsp (DWORD ServiceSessionID, WORD Response);
```

Parameters

[ServiceSessionID] Identifies the service session that is going to be terminated.

[Response] Identifies the response value (RESPONSE_OK or RESPONSE_ERROR).

Return Values

If no error occurs it returns RESULT_OK. Otherwise, it returns one of the following error codes.

INVALID_SESSION_ID: This value indicates that the user specified an invalid service session identifier in the [ServiceSessionID] parameter.

DMIF_INTERNAL_ERROR: This value indicates that an error, possibly related to memory allocation, occurred in the DMIF layer. In this case, the user must close the application.

7.2.2.4.1.8 DAI_ServiceDetachCnf

This function is called by the DMIF in reply to the original DAI_ServiceDetachReq function. The [Response] parameter is the one specified by the peer user in the DAI_ServiceDetachRsp function. In case of a positive response, DMIF frees all resources used for this service session.

```
void DAI_ServiceDetachCnf (DWORD ServiceSessionID, WORD Response);
```

Parameters

[ServiceSessionID] Identifies the service session that is going to be terminated.

[Response] Identifies the response value (RESPONSE_OK or RESPONSE_ERROR).

7.2.2.4.2 Channel primitives

7.2.2.4.2.1 DAI_ChannelAddReq

This function is called by the application to request the establishment of *one* end-to-end transport channel in the context of a particular service session identified by the `[ServiceSessionID]` value. Each channel is requested by providing an optional QoS descriptor, a direction parameter for the Data flow (`UPSTREAM, DOWNSTREAM, BIDIRECTIONAL`) and optional additional information for that requested channel in the `[uuData]` field. This information is opaque to the DMIF layer and is only consumed by the peer user. In case of an `UPSTREAM` direction, that means the Data flow originates from the side that requested the channel establishment, the `[QoS_Descriptor]` parameter should contain valid Data. In case of a `DOWNSTREAM` direction it should contain zero values.

```
int DAI_ChannelAddReq (      DWORD   ServiceSessionID,   DWORD  direction,
                             const    struct    QoS_Descriptor*    QoS_Info,
                             const BYTE* uuData, DWORD uuDataLength);
```

Parameters

[ServiceSessionID] Identifies the service session identifier.

[direction] Indicates the Data flow direction (UPSTREAM, DOWNSTREAM or BIDIRECTIONAL).

[uuData] Points to the user Data in case the user needs to transfer additional information for the requested channel. If no Data is specified, its value should be equal to NULL.

[uuDataLength] The bytes of the provided Data. If no Data is defined, its value should be equal to zero.

[QoS_Info] The QoS_Descriptor structure is used to specify the QoS requirements of the requested channel.

The QoS_Descriptor is described in Annex A.5

The decision whether to use existing resources or to create a new channel, depends on these values. If there are no specific QoS requirements it is allowed to set the parameter to NULL.

Return Values

If no error occurs it returns RESULT_OK. Otherwise, it returns one of the following error codes.

INVALID_SESSION_ID: This value indicates that the user specified an invalid service session identifier in the [ServiceSessionID] parameter.

NO_MORE_CHANNELS_SUPPORTED: This value indicates that a memory allocation failure happened.

QOS_UNSPECIFIED: This value indicates that the user did not define the [QoS_Info] parameter.

NOT_SUPPORTED_QOS: This value indicates that the user defined a QoS which is not feasible for requesting a channel within a specific service.

7.2.2.4.2.2 DAI_ChannelAddInd

This function is called by the DMIF layer to inform the application that the establishment of one end-to-end transport channel is requested by the peer entity. This channel is requested in the context of a particular service session identified by the `[ServiceSessionID]` value. DMIF assigns a local, unique value `[ChannelID]` that identifies the channel that will transport and deliver the application Data. The application should save this value and always refer to it in subsequent interactions with the DMIF concerning this channel, like Data transfer.

```
void DAI_ChannelAddInd (     DWORD ServiceSessionID, DWORD ChannelID,
                             DWORD Direction, const BYTE* uuData,
                             DWORD uuDataLength,
                             const struct QoS_Descriptor* qos_Info);
```

Parameters

[ServiceSessionID] Identifies the service session identifier.

[ChannelID] Identifies the channel that will deliver the application Data.

[Direction] Indicates the Data flow direction (UPSTREAM, DOWNSTREAM or BIDIRECTIONAL).

[uuData] In case the user specified additional Data in the DAI_ChannelAddReq function, this parameter points to it.

[uuDataLength] In case the [uuData] parameter contains valid Data this parameter identifies the bytes of the received Data. Otherwise, it is equal to zero.

[QoS_Info] See definition in DAI_ChannelAddReq function.

7.2.2.4.2.3 DAI_ChannelAddRsp

This function is called by the application as a Response to the `DAI_ChannelAddInd` function. The `[ChannelID]` value identifies the channel and should be equal to the `[ChannelID]` value provided by the `DAI_ChannelAddInd` function. If the user accepts the channel establishment the `[Response]` parameter should be equal to `RESPONSE_OK`, otherwise to `RESPONSE_ERROR`. In case of a `DOWNSTREAM` direction, that means the data flow originates from the side that received the channel establishment indication via the `DAI_ChannelAddInd` function, the `[QoS_Info]` parameter should contain valid data. In case of an `UPSTREAM` direction, it should contain zero values. The user may also provide additional information in the `[uuData]` field for the requested channel. This information is opaque to the DMIF layer and is only consumed by the peer user requesting the channel establishment.

```
int DAI_ChannelAddRsp (      DWORD ServiceSessionID, DWORD ChannelID,
                             WORD Response, const struct
                             QoS_Descriptor* QoS_Info,
                             const BYTE* uuData, DWORD uuDataLength);
```

Parameters

[ServiceSessionID] Identifies the service session identifier.

[ChannelID] Identifies the channel that will deliver the application Data.

[Response] Identifies the Response value (RESPONSE_OK or RESPONSE_ERROR).

[QoS_Info] See definition in DAI_ChannelAddReq function.

[uuData] Points to the user data in case the user needs to transfer additional information. If no data is specified, its value should be equal to NULL.

[uuDataLength] The bytes of the provided Data. If no Data is defined, its value should be equal to zero.

Return Values

If no error occurs it returns RESULT_OK. Otherwise, it returns one of the following error codes.

INVALID_SESSION_ID: This value indicates that the user specified an invalid service session identifier in the [ServiceSessionID] parameter.

INVALID_CHANNEL_ID: This value indicates that the user specified an invalid channel handle in the [ChannelID] parameter.

NO_MORE_CHANNELS_SUPPORTED: This value indicates that a memory allocation failure occured.

QOS_UNSPECIFIED: This value indicates that the user did not define the [QoS_Info] parameter.

NOT_SUPPORTED_QOS: This value indicates that the user defined a QoS which is not feasible for requesting a channel within a specific service.

DMIF_INTERNAL_ERROR: This value indicates that an error, possibly related to memory allocation, occurred in the DMIF layer. In this case, the user must close the application.

7.2.2.4.2.4 DAI_ChannelAddCnf

This function is called by the DMIF layer to reply to the original `DAI_ChannelAddReq` function. The `[Response]` parameter indicates if the requested transport channel is established successfully or not. In case of success, DMIF assigns a locally significant unique value `[ChannelID]` that identifies the channel that will deliver the application data. The application should save this value and always refer to it in subsequent interactions with the DMIF concerning this channel, like data transfer.

```
void DAI_ChannelAddCnf (  DWORD ServiceSessionID, DWORD ChannelID,
                          WORD Response,
                          const BYTE* uuData, DWORD uuDataLength);
```

Parameters

[ServiceSessionID] Identifies the service session identifier.

[ChannelID] Identifies the channel that will deliver the application Data.

[Response] Identifies the response value (RESPONSE_OK or RESPONSE_ERROR).

[uuData] If the user specified additional data in the DAI_ChannelAddRsp function, this parameter points to them.

[uuDataLength] If the [uuData] parameter contains valid data, this parameter identifies the bytes of the received data. Otherwise, it is equal to zero.

7.2.2.4.2.5 DAI_ChannelDeleteReq

This function is called by the application to release a transport channel identified by the `[ChannelID]` parameter.

```
int DAI_ChannelDeleteReq (      DWORD ChannelID, WORD Reason);
```

Parameters

[ChannelID] Identifies the channel that is going to be released.

[Reason] Identifies the reason for the channel release. Default value is REASON_OK.

Return Values

If no error occurs it returns RESULT_OK. Otherwise, it returns one of the following error codes.

INVALID_CHANNEL_ID: This value indicates that the user specified an invalid channel handle in the [ChannelID] parameter.

7.2.2.4.2.6 DAI_ChannelDeleteInd

This function is called by the DMIF to notify the application that the release of a transport channel, identified by the `[ChannelID]` parameter, is requested by the peer entity. The `[Reason]` parameter is the one specified by the peer user in the `DAI_ChannelDeleteReq` function.

```
void DAI_ChannelDeleteInd (DWORD ChannelID, WORD Reason);
```

Parameters

[ChannelID] Identifies the channel that is going to be released.

[reason] Identifies the reason for the channel release. The default value is REASON_OK.

7.2.2.4.2.7 DAI_ChannelDeleteRsp

This function is called by the application as a response to the `DAI_ChannelDeleteReq` function. In case of a positive response the application should stop delivering data over the indicated channel.

```
int DAI_ChannelDeleteRsp (      DWORD ChannelID, WORD Response);
```

Parameters

[ChannelID] Identifies the channel that is going to be released.

[Response] Identifies the response value (RESPONSE_OK or RESPONSE_ERROR).

Return Values

If no error occurs it returns RESULT_OK. Otherwise, it returns one of the following error codes.

INVALID_CHANNEL_ID: This value indicates that the user specified an invalid channel handle in the [ChannelID] parameter.

DMIF_INTERNAL_ERROR: This value indicates that an error, possibly related to memory allocation, occurred in the DMIF layer. In this case, the user must close the application.

7.2.2.4.2.8 DAI_ChannelDeleteCnf

This function is called by the DMIF to reply to the original `DAI_ChannelDeleteReq` function. The `[Response]` parameter is the one specified by the peer user in the `DAI_ChannelDeleteRsp` function. In case of a positive response, the application should stop delivering data over the channel indicated by the `[ChannelID]` parameter.

```
void DAI_ChannelDeleteCnf (DWORD ChannelID, WORD Response);
```

Parameters

[ChannelID] Identifies the channel that is going to be released.

[Response] Identifies the response value (RESPONSE_OK or RESPONSE_ERROR).

7.2.2.4.3 UserCommand primitives

7.2.2.4.3.1 DAI_UserCommandReq

This function is called by the application to notify the peer entity that it may start sending data. The `[ChannelID]` parameter identifies the channel that will receive the data from the peer entity. The content is opaque to DMIF.

```
int DAI_UserCommandReq (DWORD ChannelID, const BYTE* uuData,
                        DWORD uuDataLength);
```

Parameters

- `[ChannelID]` Identifies the channel that will receive the data.
- `[uuData]` Pointer to the user Data. If no Data is specified, its value should be equal to NULL.
- `[uuDataLength]` The bytes of the provided Data. If no Data is defined, its value should be equal to zero.

Return Values

If no error occurs it returns RESULT_OK. Otherwise, it returns one of the following error codes.

INVALID_CHANNEL_ID: This value indicates that the user specified an invalid channel handle in the [ChannelID] parameter.

NOT_IN_DIRECTION: This value indicates that the specified channel is used only for data transmission.

DMIF_INTERNAL_ERROR: This value indicates that an error, possibly related to memory allocation, occurred in the DMIF layer. In this case, the user must close the application.

7.2.2.4.3.2 DAI_UserCommandInd

This function is called by the DMIF to inform the application that it may start sending Data in the channel identified by the `[ChannelID]` parameter. It is assumed that the peer entity is ready to receive them so the application may start sending data by calling the `DAI_SendData` function.

```
void DAI_UserCommandInd (DWORD ChannelID, const BYTE* uuData,
                         DWORD uuDataLength);
```

Parameters

[ChannelID] Identifies the channel over which the application may start sending Data.

[uuData] In case the user specified data in the DAI_UserCommandReq function this parameter points to them.

[uuDataLength] In case the [uuData] parameter contains valid data this parameter identifies the bytes of the received data. Otherwise, it is equal to zero.

7.2.2.4.4 Data primitives

7.2.2.4.4.1 DAI_SendData

This function is called by the application to start sending data in the channel identified by the `[ChannelID]` parameter. The application may start sending data only after `DAI_UserCommandInd` function is called which indicates that the peer entity is ready to receive Data.

```
int DAI_SendData (DWORD ChannelID, const BYTE* Data,
                  DWORD DataLength);
```

Parameters

[ChannelID] Identifies the channel over which the application sends data.

[Data] points to the data to be transmitted.

[DataLength] The bytes of the transmitted data.

Return Values

If no error occurs it returns RESULT_OK. Otherwise, it returns one of the following error codes.

INVALID_CHANNEL_ID: This value indicates that the user specified an invalid channel handle in the [ChannelID] parameter.

NOT_OUT_DIRECTION: This value indicates that the specified channel is used only for data reception.

RECEIVER_NOT_READY: This value indicates that the peer entity is not ready to receive data.

MAX_PACKET_SIZE: This value indicates that the given packet length is greater than the maximum supported size.

DMIF_INTERNAL_ERROR: This value indicates that an error, possibly related to memory allocation, occurred in the DMIF layer. In this case the user must close the application.

7.2.2.4.4.2 DAI_DataReceived

This function is called by the DMIF to notify the application that data is received in the channel identified by the `[ChannelID]` parameter. The `[error_flag]` parameter indicates whether the received data is valid or not.

```
void DAI_DataReceived(     DWORD ChannelID, const BYTE* Data,
                           DWORD DataLength, DWORD ErrorFlag);
```

Parameters

[ChannelID] Identifies the channel over which the application sends data.

[Data] Points to the received data.

[DataLength] The bytes of the received data.

[ErrorFlag] Indicates if the received data is corrupted or not (DATA_OK or DATA_CORRUPTED).

7.3 The Conformance ATS

7.3.1.1 Introduction

This subclause contains the conformance abstract test suites (ATSs) for the DMIF Application Interface and for the DMIF Signalling protocol as specified in the ISO/IEC 14496- 6. A common test method has been identified; with reference to it each ATS specifies the test coverage and the test cases. Each test case contains the test purpose, a preamble and the test procedures. The test procedures are described in text and use arrow graphs to specify the message exchanges.

7.3.1.2 Test Method

For conformance testing of the DMIF Application Interface and of the DMIF Signalling protocol, the Remote Test Method as defined in ISO/IEC 9646-parts 1-2 is used. The characteristics of the Remote Test Method, shown in Figure 16, are as follows:

- It has only one Point of Control and Observation (PCO) between the Lower Tester and the physical layer.

- There are no requirements for the Implementation Under Test (IUT).

- This method can be called a “Black Box” test.

- It has a limit in test coverage because it can observe the behaviours of IUT

only through a lower interface.

The Lower Tester (LT) is connected to the System Under Test (SUT) through the service provider. It simulates the operation of the peer end system (Note: In some cases the peer end system is a broadcast network or a local storage media). The Service Provider supports the protocol stacks and/or a core network for communication between the LT and the Service elements in the SUT. The Upper Tester(UT) is located on the IUT and emulates the operation of applications. The remote test method does not require an UT in the SUT. However, when there are functions available in the SUT that can control the upper interface of the IUT, they should be used. In these cases, the test co-ordination procedure (TCP) between the LT and the UT is necessary. The UT and the TCP, therefore, are indicated in dotted lines in Figure 16.

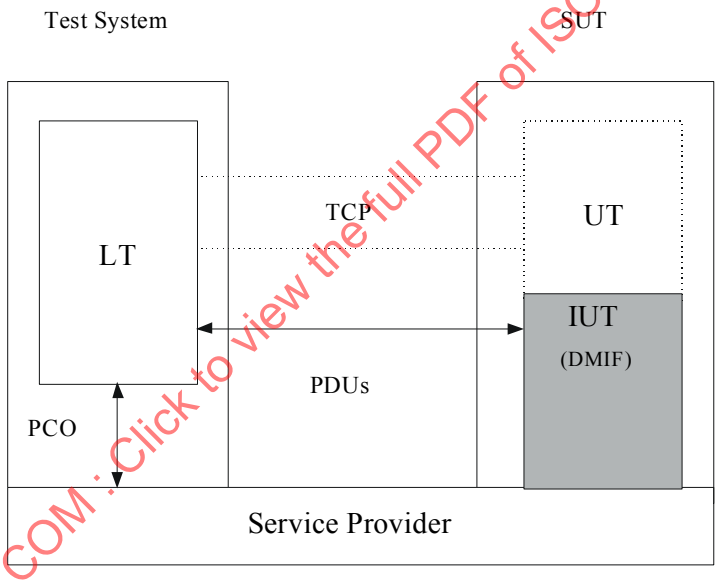


Figure 16 — The Remote Test Method

7.3.2 ATS for DAI in Remote Interactive Scenarios

7.3.2.1 Generic ATS for DAI in Remote Interactive Scenarios

7.3.2.1.1 Generic Test Environment

Figure 17 shows the function and role of the LT and the SUT in the DAI test environment for a remote interactive scenario.

The UT is required for all test cases.

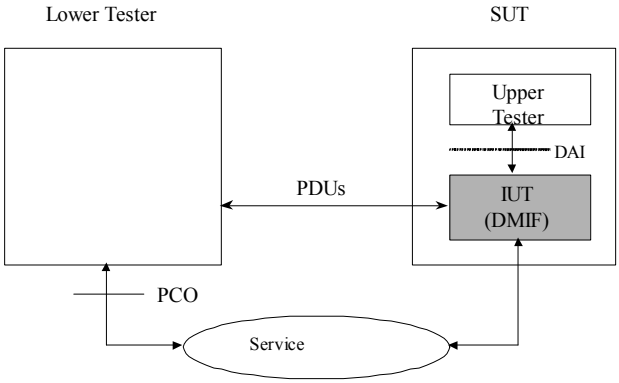


Figure 17 — The test method for the DAI test environment for a remote interactive scenario

The Upper Tester transmits and examines DAI primitives exchanged with the IUT. That is, the UT sends DAI requests and waits for the corresponding confirmations from the IUT. In some cases the UT waits for indications from the IUT and provides the corresponding responses. The UT can also generate invalid/inopportune messages to test the IUT's error handling capability.

The Lower Tester transmits and examines DS protocol messages in and out of the IUT. That is, the LT sends DS protocol messages requests and waits for the corresponding confirmations from the IUT. In some cases the LT waits for indications from the IUT and provides the corresponding responses. The DMIF Protocol Emulator can also generate invalid/inopportune messages to test the IUT's error handling capability.

The PCO is represented by the DS protocol messages at the LT.

7.3.2.1.2 Generic Test Cases

7.3.2.1.2.1 DMIF Service Primitives

The test cases described in subclause 7.2.2.4.1 are valid.

7.3.2.1.2.2 DMIF Channel primitives

The test cases described in subclause 7.2.2.4.2 are valid.

7.3.2.1.2.3 DMIF User Primitives

The test cases described in subclause 7.2.2.4.3 are valid.

7.3.2.1.2.4 DMIF Data Primitives

The test cases described in subclause 7.2.2.4.4 are valid.

7.3.3 ATS for DAI in Local Storage Scenarios

7.3.3.1 Generic ATS for DAI in Local Storage Scenarios

7.3.3.1.1 Generic Test Environment

Figure 18 shows the function and role of the LT and the SUT in the DAI test environment for a local file scenario.

The UT is required for all test cases.

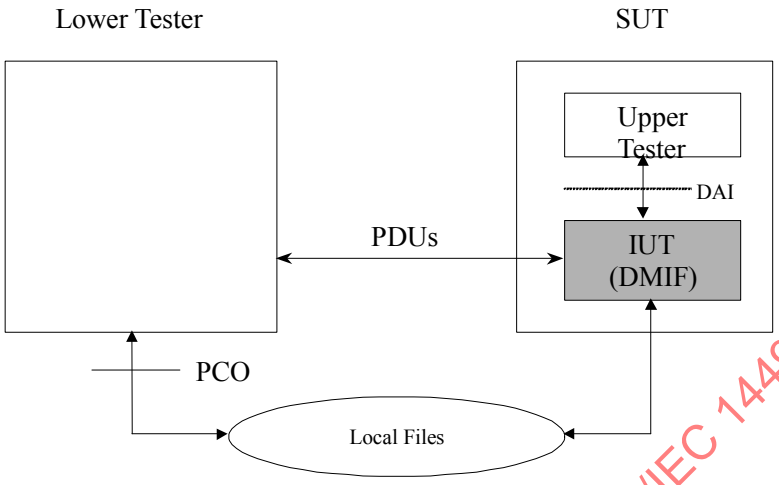


Figure 18 — The test method for the DAI test environment for a local file scenario

The Upper Tester transmits and examines DAI primitives exchanged with the IUT. That is, the UT sends DAI requests and waits for the corresponding confirmations from the IUT. In some cases the UT waits for indications from the IUT and provides the corresponding responses. The UT can also generate invalid/inopportune messages to test the IUT's error handling capability.

The LT produces and/or examines a local file depending on whether the test is for reading and/or writing to file.

The PCO is represented by the file itself.

7.3.3.1.2 Generic Test Cases

7.3.3.1.2.1 DMIF Service primitives

Table 109 — DMIF DAI Service Primitives Test Cases

Test Case #	Test Case Names	Reference to ISO 14496-6
1	Attaching a Service	10.4.1
2	Detaching a Service	10.4.3

7.3.3.1.2.1.1 Test Case 1 - Attaching a Service

Test Purpose:

Verify that a Service Attach procedure is correctly performed for Local Storage.

Test Preamble:

a) The UT requests through a DA_ServiceAttach(IN) to create a new service

b) The service is located in the local storage content

Test Procedure:

1. UT passes a DA_ServiceAttach(IN) to the IUT
2. IUT passes a DA_ServiceAttach(OUT) back to the UT

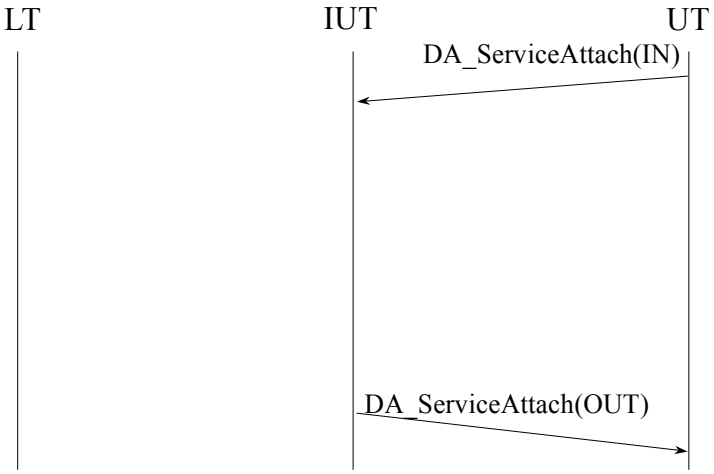


Figure 19 —Service Attach for Local Storage

Test Verdict:

Pass the test if the response to UT is:

Conditions	Observations
DA_ServiceAttach(IN) was valid	DA_ServiceAttach(OUT) has been generated consistently with response OK
DA_ServiceAttach(IN) was not valid	DA_ServiceAttach(OUT) has been generated consistently with response not OK

7.3.3.1.2.1.2 Test Case 2 - Detaching a Service

Test Purpose:

Verify that a Service Detach procedure is correctly performed for Local Storage.

Test Preamble:

a) The UT requests through a DA_ServiceDetach(IN) to detach a service

b) The service was existing and operating in the local storage session

Test Procedure:

1. UT passes a DA_ServiceDetach(IN) to the IUT
2. IUT passes a DA_ServiceDetach(OUT) back to the UT

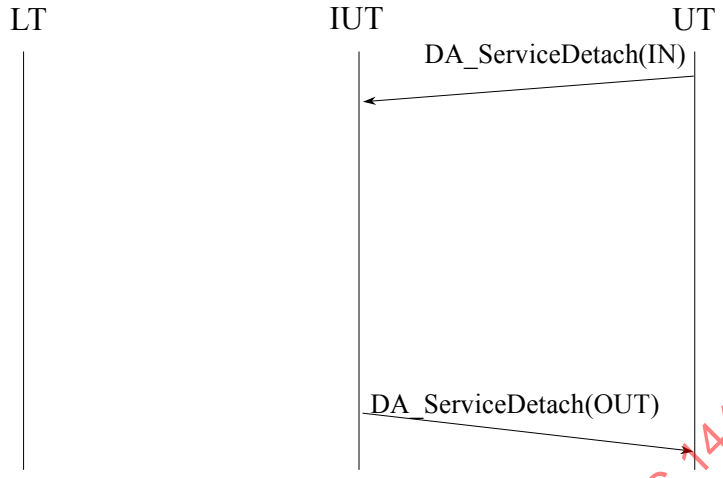


Figure 20 — Service Detach for Local Storage

Test Verdict:

Pass the test if the response to UT is:

Conditions	Observations
DA_ServiceDetach(IN) was valid	DA_ServiceDetach(OUT) has been generated consistently with response OK
DA_ServiceDetach(IN) was not valid	DA_ServiceDetach(OUT) has been generated consistently with response not OK

7.3.3.1.2.2 DMIF Channel primitives

Table 110 — DMIF DAI Service Primitives Test Cases

Test Case #	Test Case Names	Reference to ISO/IEC 14496-6
1	Adding a Channel	10.4.5
2	Deleting a Channel	10.4.7

7.3.3.1.2.2.1 Test Case 1 - Adding a Channel

Test Purpose:

Verify that a Channel Add procedure is correctly performed for Local Storage.

Test Preamble:

a) The UT requests through a DA_ChannelAdd(IN) to add a new channel to a service

b) The service is already existing and operating in the local storage session

Test Procedure:

1. UT passes a DA_ChannelAdd(IN) to the IUT
2. IUT passes a DA_ChannelAdd(OUT) back to the UT

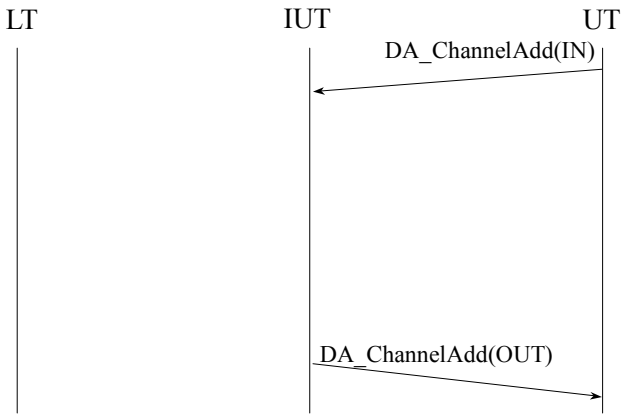


Figure 21 — Channel Add for Local Storage

Test Verdict:

Pass the test if the response to UT is:

Conditions	Observations
DA_ChannelAdd(IN) was valid	DA_ChannelAdd(OUT) has been generated consistently with response OK
DA_ChannelAdd(IN) was not valid	DA_ChannelAdd(OUT) has been generated consistently with response not OK

7.3.3.1.2.2.2 Test Case 3 - Deleting a Channel

Test Purpose:

Verify that a Service Detach procedure is correctly performed for Local Storage.

Test Preamble:

a) The UT requests through a DA_ChannelDelete(IN) to delete a channel

b) The channel was existing and operating in the local storage session

Test Procedure:

1. UT passes a DA_ChannelDelete(IN) to the IUT
2. IUT passes a DA_ChannelDelete(OUT) back to the UT

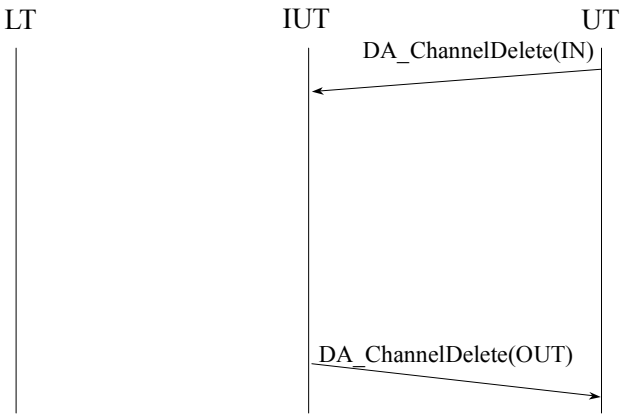


Figure 22 — Channel Delete for Local Storage

Test Verdict:

Pass the test if the response to UT is:

Conditions	Observations
DA_ChannelDelete(IN) was valid	DA_ChannelDelete(OUT) has been generated consistently with response OK
DA_ChannelDelete(IN) was not valid	DA_ChannelDelete(OUT) has been generated consistently with response not OK

7.3.4 ATS for DAI in Broadcast Scenarios

7.3.4.1 Generic ATS for DAI in Broadcast Scenarios

7.3.4.1.1 Generic Test Environment

Figure 23 shows the function and role of the LT and the SUT in the DAI test environment for a broadcast scenario.

The UT is required for all test cases.

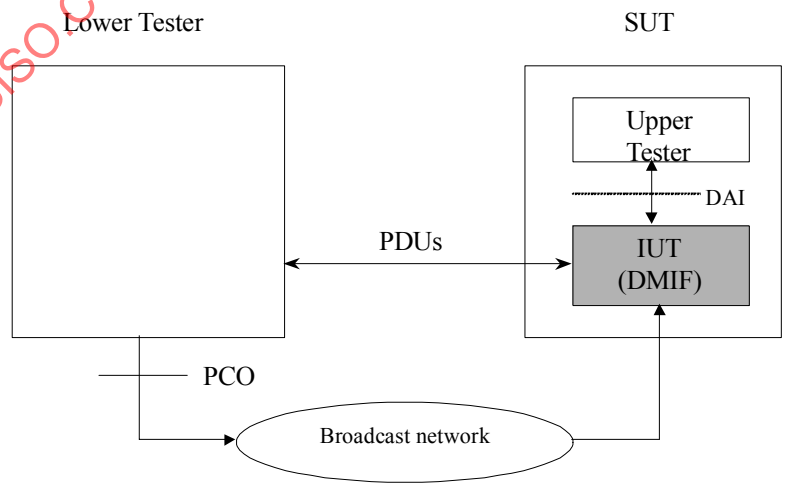


Figure 23 — The test method for the DAI test environment for a broadcast scenario

The Upper Tester transmits and examines DAI primitives exchanged with the IUT. That is, the UT sends DAI requests and waits for the corresponding confirmations from the IUT. In some cases the UT waits for

indications from the IUT and provides the corresponding responses. The UT can also generate invalid/inopportune messages to test the IUT's error handling capability.

The LT broadcasts streams to IUT.

The PCO is represented by the broadcast traffic emitted by the LT.

7.3.4.1.2 Generic Test Cases

7.3.4.1.2.1 DMIF Service primitives

Table 111 — DMIF DAI Service Primitives Test Cases

Test Case #	Test Case Names	Reference to ISO/IEC 14496-6
1	Attaching a Service	10.4.1
2	Detaching a Service	10.4.3

7.3.4.1.2.1.1 Test Case 1 - Attaching a Service

Test Purpose:

Verify that a Service Attach procedure is correctly performed for Broadcast.

Test Preamble:

a) The UT requests through a DA_ServiceAttach(IN) to create a new service

b) The service is located in the broadcast content

Test Procedure:

1. UT passes a DA_ServiceAttach(IN) to the IUT
2. IUT passes a DA_ServiceAttach(OUT) back to the UT

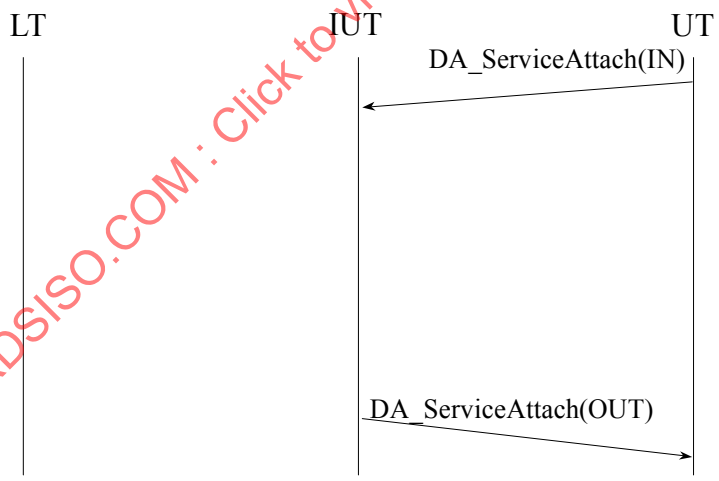


Figure 24 — Service Attach for Broadcast

Test Verdict:

Pass the test if the response to UT is:

Conditions	Observations
DA_ServiceAttach(IN) was valid	DA_ServiceAttach(OUT) has been generated consistently with response OK
DA_ServiceAttach(IN) was not valid	DA_ServiceAttach(OUT) has been generated consistently with response not OK

7.3.4.1.2.1.2 Test Case 2 - Detaching a Service

Test Purpose:

Verify that a Service Detach procedure is correctly performed for Broadcast.

Test Preamble:

a) The UT requests through a DA_ServiceDetach(IN) to detach a service

b) The service was existing and operating in the broadcast session

Test Procedure:

1. UT passes a DA_ServiceDetach(IN) to the IUT
2. IUT passes a DA_ServiceDetach(OUT) back to the UT

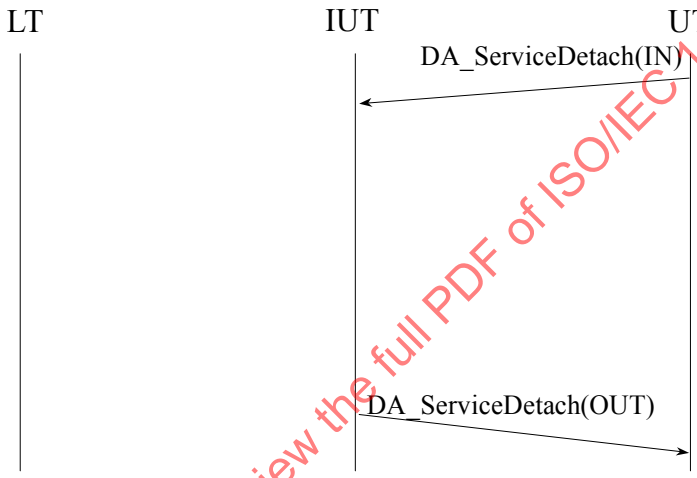


Figure 25 — Service Detach for Broadcast

Test Verdict:

Pass the test if the response to UT is:

Conditions	Observations
DA_ServiceDetach(IN) was valid	DA_ServiceDetach(OUT) has been generated consistently with response OK
DA_ServiceDetach(IN) was not valid	DA_ServiceDetach(OUT) has been generated consistently with response not OK

7.3.4.1.2.2 DMIF Channel primitives

Table 112 — DMIF DAI Service Primitives Test Cases

Test Case #	Test Case Names	Reference to ISO/IEC 14496-6
1	Adding a Channel	10.4.5
2	Deleting a Channel	10.4.7

7.3.4.1.2.2.1 Test Case 1 - Adding a Channel

Test Purpose:

Verify that a Channel Add procedure is correctly performed for Broadcast.

Test Preamble:

a) The UT requests through a DA_ChannelAdd(IN) to add a new channel to a service

b) The service is already existing and operating in the broadcast session

Test Procedure:

1. UT passes a DA_ChannelAdd(IN) to the IUT
2. IUT passes a DA_ChannelAdd(OUT) back to the UT

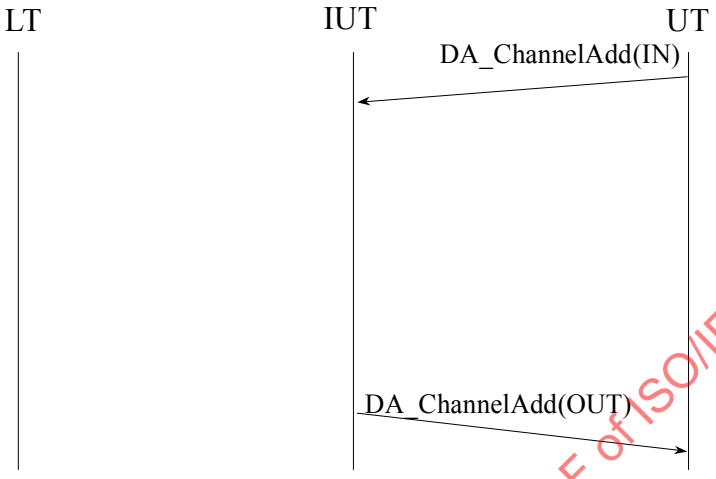


Figure 26 — Channel Add for Broadcast

Test Verdict:

Pass the test if the response to UT is:

Conditions	Observations
DA_ChannelAdd(IN) was valid	DA_ChannelAdd(OUT) has been generated consistently with response OK
DA_ChannelAdd(IN) was not valid	DA_ChannelAdd(OUT) has been generated consistently with response not OK

7.3.4.1.2.2.2 Test Case 3 - Deleting a Channel

Test Purpose:

Verify that a Service Detach procedure is correctly performed for Broadcast.

Test Preamble:

a) The UT requests through a DA_ChannelDelete(IN) to delete a channel

b) The channel was existing and operating in the broadcast session

Test Procedure:

1. UT passes a DA_ChannelDelete(IN) to the IUT
2. IUT passes a DA_ChannelDelete(OUT) back to the UT

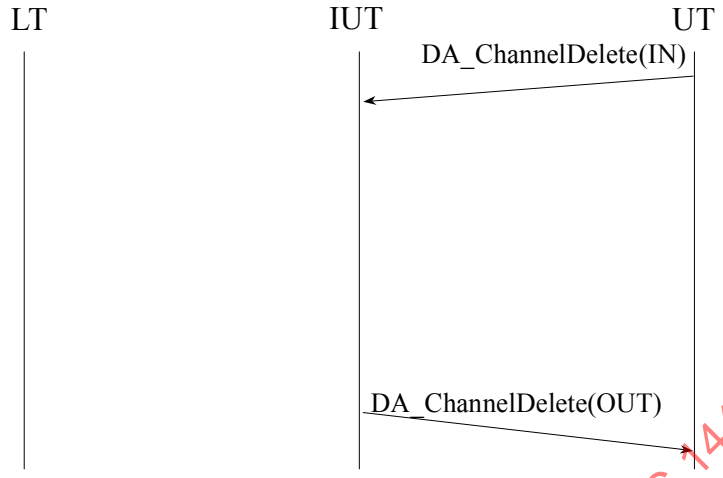


Figure 27 — Channel Delete for Broadcast

Test Verdict:

Pass the test if the response to UT is:

Conditions	Observations
DA_ChannelDelete(IN) was valid	DA_ChannelDelete(OUT) has been generated consistently with response OK
DA_ChannelDelete(IN) was not valid	DA_ChannelDelete(OUT) has been generated consistently with response not OK

8 SNHC

8.1 Introduction

The decoding functionalities for SNHC in MPEG-4 Version 2 are Body Animation and 3D Mesh Coding (3DMC) with CGD. The Version 2 SNHC functionalities involve both downloading (write once, render may) and streaming (write-render-many) modes. When Body Animation models or more general models under 3DMC are downloaded, the user will care about overall latency to receive a freestanding lump of a 3D model that can be usefully rendered. Thus the model must be complete in terms of connectivity (topology), shape (vertex geometry), and appearance (color, texture if any including texture coordinates, shading normals, etc.) before rendering can begin. With streaming models (that may be animated, partitioned for incremental rendering, or enriched hierarchically with expanding detail over time), the user will be sensitive to the time taken to receive each new rendition or part of the ultimate model without loosing patience. Decoder rates, within practical limits of the platform technology in which MPEG-4 will be deployed, should be chosen to keep pace with the arrival of successive states in the progress of model animation or build-up.

Applications for Body Animation include both broadcast and interactive sessions analogous to the applications for Face Animation. Consequently, many of the profiling and conformance issues associated with Body Animation Parameters have been addressed by Face Animation in the MPEG-4 Version 1 work. BAPs are more numerous than FAPs, so the profile-at-level points for BAPs will involve typically somewhat higher bitrates and different quantization factors depending on the range of body movements compared to face, with underlying decoder functionalities similar to FAPs. The size of the Body Animation Tables that associate animated joint angles or segment lengths with control points in skin and clothing mesh models will scale with the numbers of BAPs used, though not with the complexity of the body mesh models. BAP profiling should be designated separately from mesh model delivery. MPEG-4 does not generally standardize face/body or more general models, so no profiling of specific models is relevant.

The rate at which models of varied complexity can be delivered and decoded is a concern in transmitting compressed models. 3DMC conformance is examined in this way. Body (like Face) Animation involves the option to download a body/clothing model with a potentially large range in fidelity and complexity, tied to the

kinematic structure controlled by Body Animation Parameters (BAP). Face and body model downloads are designed to harness the Version 2 capability for compressed triangular or polygonal meshes of 3DMC to customize surface shape and appearance. Since body models are just (H-Anim compliant) instances of more general models to be deployed with MPEG-4, the profile and level considerations of 3DMC should subsume the download of face and body models. This framework provides support for Face Animation from Version 1 to utilize 3DMC in Version 2 as well. Thus FBA applications treat animation and model delivery profiling separately.

8.1.1 Purpose & Scope

The purpose of this subclause is to provide a concise and complete description including sample data annexes and electronic test data with which to certify MPEG-4 SNHC-compliant decoders. The scope of this document is to provide additions to Part 4 for the SNHC functionalities of Face and Body Animation (FBA) and 3D Mesh coding (3DMC) that are covered in Version 1 by profiles in the Visual Part of the specification. To the extent that these capabilities are invoked by profiles at prescribed levels of performance, a decoder for a given functionality shall conform to the requirements of this document.

In specific cases, profile and level requirements for SNHC functionalities specified in ISO/IEC 14496 can depend on the unified application of related features in Part 1 of this specification. In particular, SNHC functionalities can be deployed with elements of Part 1 of ISO/IEC 14496 including elementary streams, scene description and composition supported by the Binary Interchange Format for Scenes (BIFS).

An example is the downloading of Face and Body Definition Parameters to describe a specific face/body model in preparation for subsequent animation of that model by FBA Parameters. In such cases, decoders and other terminal resources shall conform to this document to the extent that simultaneous performance and functional capabilities are invoked in the Visual and Systems parts.

8.1.2 Intended Use of Decoders

SNHC-related decoders are intended for use in stand-alone applications with or without Systems BIFS depending on the Systems Profile definitions. For example, FBA can be used in a broadcast application without any terminal utilization of downloaded face models. In such a case, the decoder is expected to be able to make a connection to a broadcast at any time and is to be tested for specified functionalities and performance without connection to Systems.

FBA can also be used to download a specific (possibly textured) model of a polygonal mesh representing a talking head and subsequently to animate the feature control points of such a customized model. Then testing should be conducted with the specified decoder and BIFS functionality.

8.1.3 What Is To Be Tested

The intent of this document is generally to provide the following requirements for testing and the resources included in or referenced by this document to the extent applicable.

A. Specification of conditions in the decoding of compliant bitstream testing for:

1. Exercising all functionalities and modes of decoding to produce expected results,
2. Exercising the full range of bitstream rates, frame rates, skip frame, quantization step size,
3. Exercising min/max and representative values of bitstream parameters that stress computational limits and conditional execution of algorithms,
4. Achieving profile/level performance points in real-time or non-real-time as applicable with specified content in compliant bitstreams, and
5. Fully utilizing any required terminal resources at specified levels (e.g. memory for data tables/models, conditions of rendering such as achieved picture area, update rates).

B. Provision of required test data in an accessible format with which hardware and software manufacturers shall conduct specified testing.

8.2 Body Animation

8.2.1 Simple FBA Profile

The following profile structure is recommended for FBA in ISO/IEC 14496-1:2001 based on extending the Profile @ Level structure for Face Animation from ISO/IEC 14496-1:1999. This is linked to the assumption that Face nodes will be subsumed under the Body node in ISO/IEC 14496-1:2001, and that BIFS-related

Profiles will be adjusted as needed accordingly. All ISO/IEC 14496-2 FBA decoders (for all object types) are required to generate at their output a humanoid model including all the feature points and joints defined in ISO/IEC 14496-2, even if some of the features points and joints will not be affected by any information received from the encoder.

Level 1:

- number of objects: 1,
- The total FBA decode frame-rate in the bitstream shall not exceed 72 Hz,
- The decoder shall be capable of a humanoid model rendering update of at least 15 Hz, and
- Maximum bitrate 32 kbit/s.

Level 2:

- maximum number of objects: 4,
- The FBA decode frame-rate in the bitstream shall not exceed 72 Hz (this means that the FBA decode framerate is to be shared among the objects),
- The decoder shall be capable of rendering the humanoid models with the update rate of at least 60Hz, sharable between humanoids, with the constraint that the update rate for each individual humanoid is not required to exceed 30Hz, and
- Maximum bitrate 64 kbit/s.

8.2.2 FBA Conformance Points

8.2.2.1 Covered Functionalities

8.2.2.1.1 Face and Body Animation (FBA)

FBA in MPEG-4 Version 1 includes the specification of highly efficient coding of animation parameters whose decoding can drive an unlimited range of possible non-normative face models. The functionalities provided include arithmetic and DCT coding of a large collection of FAPs for accurate speech articulation, as well as viseme and expression parameters to code specific speech configurations of the lips and the mood of the speaker. This core animation parameter coding is structured for baseline conformance testing without normative dependence on Systems. The core FBA capability can be used without or with other features of FBA supported in Systems BIFS, again depending on the application.

The Systems BIFS features supporting FBA include:

1. the Face/Body Definition Parameters (FDP/BDP) in BIFS (model data downloadable to configure a baseline face model pre-stored in the terminal into a particular face/body before FAP/BAP decoding, or to install a specific face/body model at the beginning of a session along with the information about how to animate it),
2. the Face/Body Deformation Table (FaceDefTable/BodyDefTable) within FDP/BDPs (downloadable functional mapping from incoming FAP/BAPs to feature control points in the face mesh that provides piecewise linear weightings of incoming FAP/BAPs for controlling face/body movements).

As applicable, the Systems functionality above shall be tested for decoder conformance with specified functionalities and performance from the profiles of the Visual and Systems parts of the specification. Simultaneous testing shall be conducted to the extent that Visual Profiles require the simultaneous achievement of specified decoding throughput, the exercise of BIFS capabilities to customize and animate the model, and/or the display of specific characteristics in the final face rendering of non-normative models downloaded under BIFS.

8.2.2.2 Description/References on Conformance Definitions

8.2.2.2.1 FBA

####### 8.2.2.2.1.1 Introduction

The required decoder functionalities to be achieved in testing are specified in the following detailed subclauses of the Visual Part 2 and Systems Part 1 of the MPEG-4 specification:

The FBA specification is defined in ISO/IEC 14496-1 and ISO/IEC 14496-2. This clause is intended to facilitate finding various parts of specification. As a rule of thumb, FAP/BAP specification is found in the part 2, and FDP/BDP specification in the part 1. However, this is not a strict rule. For an overview of FAP/BAPs and their interpretation, read clauses “6.1.5.2 Facial animation parameter set”, “6.1.5.3 Facial animation parameter units”, “6.1.5.4 Description of a neutral face” as well as the Table C.1 (Annex C) of ISO/IEC 14496-2. The viseme parameter is documented in clause “7.12.3 Decoding of the viseme parameter fap 1” and the Table C.5 (Annex C) of ISO/IEC 14496-2. The expression parameter is documented in clause “7.12.4 Decoding of the expression parameter fap 2” and the Table C.3 of ISO/IEC 14496-2. FAP/BAP bitstream syntax is found in clause “6.2.10 Face Object”, semantics in “6.3.10 Face Object”, and clause “7.12 Face object decoding” of ISO/IEC 14496-2 explains in more detail the FAP/BAP decoding process. FAP/BAP masking and interpolation is explained in clauses “6.3.11.1 Face Object Plane”, “7.12.1.1 Decoding of FAP/BAPs”, “7.12.5 FAP/BAP masking” of ISO/IEC 14496-2. The FIT interpolation scheme is documented in clause “7.2.5.3.2.4 FIT” of ISO/IEC 14496-1. The FDPs and their interpretation are documented in clause “7.2.5.3.2.6 FDP” of ISO/IEC 14496-1. In particular, the FDP feature points are documented in the Annex C of ISO/IEC 14496-2.

8.2.2.2.1.2 Profile and Level Definitions

The profile and levels that FBA are defined in clause 9 of ISO/IEC 14496-2. These profiles are:

1. Simple FBA Profile,

Each of these profiles has two levels; a compliant bitstream of a given profile-and-level may be called an “ISO/IEC 14496-2 *Profile@Level* bitstream”.

8.2.3 FBA Testing Conditions

8.2.3.1 Description of Test Data

8.2.3.1.1 CD-ROM & Textual Test Data

SNHC conformance test data is supplied in the MP4 format in the electronic attachment of this document.

8.2.3.1.2 FBA

In the facial animation subclauses of this part of ISO/IEC 14496, except where stated otherwise, the following terms are used for practical purposes:

The term *'bitstream'* means ISO/IEC 14496 facial animation bitstream.

The term *'decoder'* means ISO/IEC 14496 facial animation decoder, i.e. an embodiment of the decoding process specified by ISO/IEC 14496-2. The term *'verifier'* means an ISO/IEC 14496 facial animation bitstream verifier, i.e., a process by which it is possible to test and verify that all the requirements specified in ISO/IEC 14496-2 are met by the bitstream.

If any statement stated in this subclause accidentally contradicts a statement or requirement defined in ISO/IEC 14496-2, the text of ISO/IEC 14496-2 prevails. The following subclauses specify the normative tests for verifying conformance of facial animation bitstreams and facial animation decoders. Those normative tests make use of test data (bitstream test suites) provided as an electronic annex to this document, and of a software verifier specified in ISO/IEC 14496-5 with source code available in electronic format.

8.2.3.1.2.1 Definition of FBA bitstream conformance

An ISO/IEC 14496 facial animation bitstream is a bitstream that implements the specification defined by the normative clauses of ISO/IEC 14496-2 (including all normative annexes of ISO/IEC 14496-2).

A compliant bitstream shall meet all the requirements and implement all the restrictions defined in the generic syntax defined by the ISO/IEC 14496-2 specification, including the restrictions defined in clause 9 of ISO/IEC 14496-2 for the profile-and-level specified for the bitstream.

A compliant bitstream of a given profile-and-level may be called an “ISO/IEC 14496-2 *Profile@Level* bitstream” or simply a “*Profile@Level* bitstream” (e.g. an MP@ML bitstream).

The profile_and_level_indication shall be one of the valid codes defined in clause 9 of ISO/IEC 14496-2. The profile-and-level derived from the profile_and_level_indication indicates that additional restrictions and constraints have been applied to several syntactic and semantic elements, as defined in clause 9 of ISO/IEC 14496-2.