INTERNATIONAL STANDARD

ISO/IEC 14496-4

First edition
2000-12-15

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 14496-4:2000(E)

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.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 3.

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this part of ISO/IEC 14496 may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

International Standard 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*.

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)*

Annexes A to D of this part of ISO/IEC 14496 are for information only.

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.

The text of ISO/IEC 14496-4 and the electronic attachments to this International Standard are provided on four CD-ROMs. All test sequences and bitstreams mentioned in the text of the standard are on these CD-ROMs.

— CD 1 contains all ISO/IEC 14496-1 and ISO/IEC 14496-2 sequences as well as those ISO/IEC 14496-3 which are not included on CD 2, CD 3 and CD 4.

— CD 2 contains ISO/IEC 14496-3 audio AAC lc, AAC ltp, AAC main and Twin-VQ sequences.

— CD 3 contains ISO/IEC 14496-3 audio AAC scalable and original sine-sweep sequences.

— CD 4 contains ISO/IEC 14496-3 audio AAC ssr sequences and International Standard ISO/IEC 14496-4.

Each CD-ROM contains a text file cdX.sum with a list of file names followed on the next line by a checksum, output of the Unix utility “sum”. This allows the integrity of zip files to be checked.

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

1 General

1.1 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.

1.2 Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this part of ISO/IEC 14496. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, 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 normative documents indicated below. For undated references, the latest edition of the normative document referred to applies. Members of ISO and IEC maintain registers of currently valid International Standards.

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

ISO 8859-1:1987*, Information processing - 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 908:1987, *Compact disc 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, *Information technology - Open Systems Interconnection - Conformance testing methodology and framework - Part 1: General concepts*.

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

ISO/IEC 9646-7, *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 11172-4:1993, *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(1995)|ISO/IEC 13818-1:1996, *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 14496-1:1999, *Information technology - Coding of audio-visual objects - Part 1: Systems*.

ISO/IEC 14496-2:1999, *Information technology - Coding of audio-visual objects - Part 2: Visual*.

ISO/IEC 14496-3:1999, *Information technology - Coding of audio-visual objects - Part 3: Audio*.

ISO/IEC 14496-6:1999, *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) Recommendation ITU-R BT.601-3, *Encoding parameters of digital television for studios*.

CCIR Volume X and XI Part 3 Recommendation ITU-R BR.648, *Recording of audio signals*.

CCIR Volume X and XI Part 3 Report ITU-R 955-2, *Satellite sound broadcasting to vehicular, portable and fixed receivers in the range 500 - 3000Mhz*.

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

ITU-T Rec. H.261 (Formerly CCITT Rec. H.261), *Video codec for audiovisual services at p x 64 kbit/s,* Geneva, 1990.

2 Technical elements

2.1 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.

2.2 Abbreviations and symbols

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.

3 Systems

3.1 Conformance Points

Figure 3-1 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.

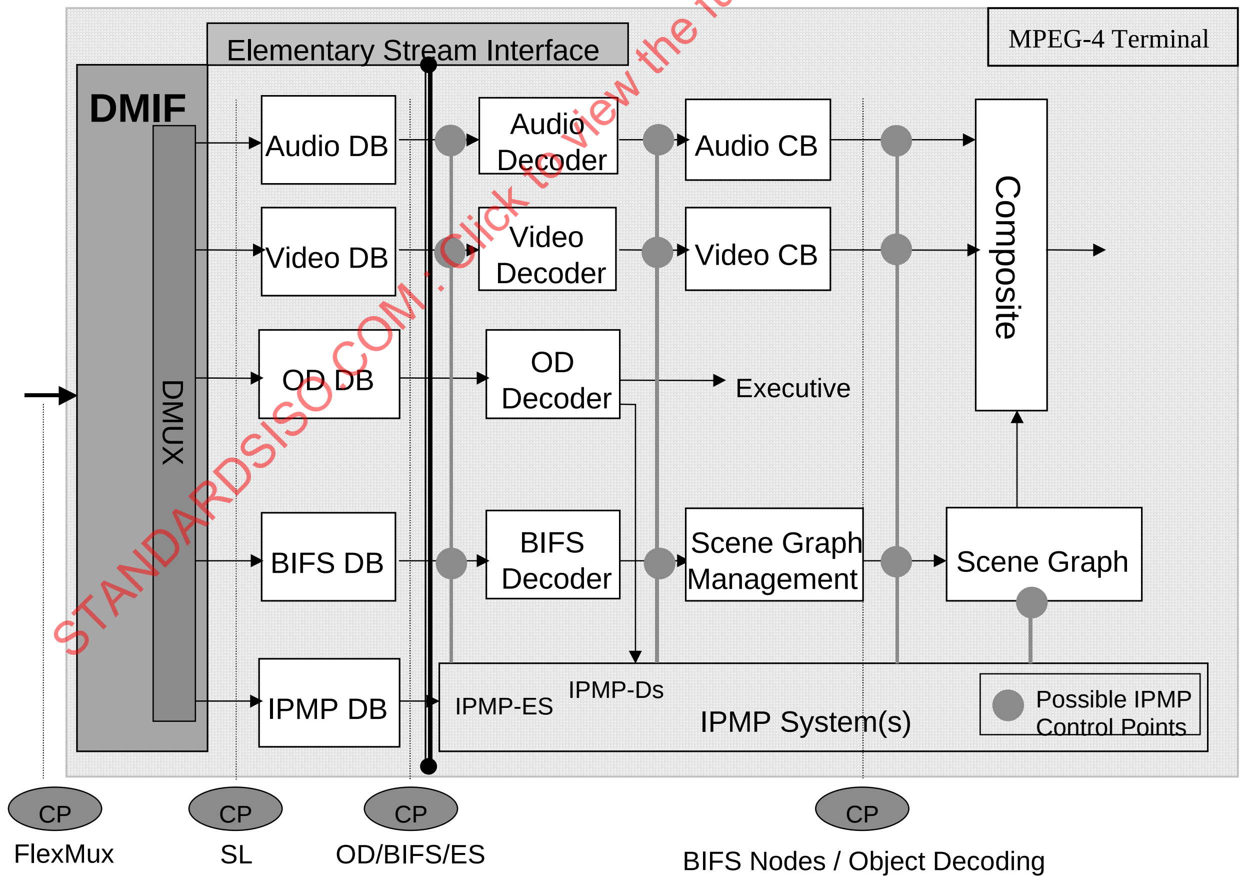

Figure 3-1 — Typical MPEG-4 terminal

On Figure 3-1, 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

3.1.1 FlexMux Conformance Point

A FlexMux conformance point is a conformance point where FlexMux streams as defined in subclause 11.2 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.

3.1.2 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.

3.1.3 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.

3.1.4 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.

3.1.5 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.

3.1.6 IPMP Conformance Point

IPMP information shall be conveyed in an MPEG-4 bitstream using the IPMP framework described in ISO/IEC 14496-1, subclauses 8.3.2 and 8.6. 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.

3.1.7 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.

3.2 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.

3.2.1 FlexMux Conformance

3.2.1.1 Conformance Requirements

FlexMux-ed bitstreams shall comply with the specifications in subclause 11.2 of ISO/IEC 14496-1

3.2.1.2 Measurement procedure

Syntax of the bitstream shall meet the requirements of subclause 11.2 of ISO/IEC 14496-1.

3.2.1.3 Tolerance

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

3.2.2 Synchronization Layer Conformance

3.2.2.1 Conformance Requirements

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

3.2.2.2 Measurement procedure

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

3.2.2.3 Tolerance

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

3.2.3 OD Conformance

3.2.3.1 Conformance Requirements

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

3.2.3.2 Measurement procedure

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

3.2.3.3 Tolerance

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

3.2.4 BIFS Conformance

3.2.4.1 Conformance Requirements

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

3.2.4.2 Measurement procedure

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

3.2.4.3 Tolerance

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

3.2.5 OCI Conformance

3.2.5.1 Conformance Requirements

OCI descriptors included in ObjectDescriptors or ES_Descriptors shall comply with ISO/IEC 14496-1 subclause 8.6. A conformant OCI bitstream shall only contain OCI events and OCI descriptors that are compliant to ISO/IEC 14496-1 subclauses 8.4 and 8.6. 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

3.2.5.2 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.

3.2.5.3 Tolerance

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

3.2.6 IPMP Conformance

3.2.6.1 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 8.3.2 and 8.6 as well as IPI Data Sets that are compliant to ISO/IEC 14496-1 subclause 8.6.8.

3.2.6.2 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 8.3.2 and 8.6.13 as well as IPI Data Sets that are parse-able.

3.2.6.3 Tolerance

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

3.2.7 Miscellaneous Conformance

3.2.7.1 Conformance Requirements

3.2.7.1.1 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.

3.2.7.1.2 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.

3.2.7.2 Measurement procedure

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

3.2.7.3 Tolerance

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

3.3 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.

3.3.1 FlexMux conformance

3.3.1.1 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.

3.3.1.2 Measurement procedure

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

3.3.1.3 Tolerance

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

3.3.2 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.

3.3.2.1 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.

3.3.2.1.1 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.

3.3.2.1.2 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.

3.3.2.2 Measurement Procedure

3.3.2.2.1 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.

3.3.2.2.1.1 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 3-1 — 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:

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

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

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 ΔTm ≤ 0, the rightmost expression will be treated as +∞.

Since this inequality must hold for all i and all j ≠ 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:

$$r_{\min} \leq r_{\max}$$

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.

3.3.2.2.1.2 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 ≠ 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 ±δ. The rounding error of two OCRs may thus accumulate to ±2δ. 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

3.3.2.2.2 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)
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.

3.3.2.3 Tolerance

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

3.3.3 OD Conformance

3.3.3.1 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.

3.3.3.1.1 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 10646:1993 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.

3.3.3.1.2 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.

3.3.3.1.2.1 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 10646:1993 (or Unicode 2.1) compliant string.

3.3.3.1.2.2 Scope of URLs in the Initial Object Descriptor

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

3.3.3.1.3 The Object Descriptor

3.3.3.1.3.1 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).

3.3.3.1.3.2 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.

3.3.3.1.4 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.

3.3.3.1.5 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.

3.3.3.1.5.1 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.

3.3.3.1.5.2 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.

3.3.3.1.5.3 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.

3.3.3.1.5.4 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.

3.3.3.1.5.5 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.

3.3.3.1.6 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.

3.3.3.1.6.1 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.

3.3.3.2 Measurement Procedure

3.3.3.3 Tolerance

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

3.3.4 BIFS Conformance

3.3.4.1 Conformance Requirements

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

3.3.4.2 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.

3.3.4.3 Tolerance

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

3.3.5 OCI Conformance

3.3.5.1 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.

3.3.5.2 Measurement procedure

This procedure is application dependant.

3.3.5.3 Tolerance

The tolerance is application dependant.

3.3.6 IPMP Conformance

3.3.6.1 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.

3.3.6.2 Measurement Procedure

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

3.3.6.3 Tolerance

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

3.3.7 Scene Graph Conformance

3.3.7.1 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.

3.3.7.2 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.

3.3.7.3 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.

3.3.8 Miscellaneous Conformance

3.3.8.1 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.

3.3.8.1.1 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.

3.3.8.1.2 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

3.3.8.1.3 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.

3.3.8.2 Measurement Procedure

3.3.8.2.1 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.

3.3.8.3 Tolerance

3.4 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.

3.4.1 Parsing Hint File Format

3.4.1.1 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”

3.4.1.2 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 3.4.1.2.1 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 3-2 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 3-2 — 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 3-3	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 3.4.1.2.1	29	40	left
<bits read>	the bits read, interpreted as a hexadecimal number	69	9	right
<blank>	blank characters for better legibility	78	2	-
<decoded value>	see subclause 3.4.1.2.2	80	N/A	left

Table 3-3 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 3-3 — 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`

3.4.1.2.1 <semantic> syntax element

As stated in Table 3-2, 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 (one exception is formed by fields of type SFBool, which cannot be broken down further). I.e., a field of type SFFloat would have to be broken down recursively until the *mantissa* and *exponent* level is reached.

3.4.1.2.2 <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.

3.4.1.2.2.1 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.

3.4.1.3 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
- 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"

3.4.1.4 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.

3.4.2 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.

[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.

3.4.2.1 Elements and their attributes

3.4.2.1.1 <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"

3.4.2.1.2 <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.

3.4.2.1.3 <scene>

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

Container:
<scene-dump>

Attributes:
(none)

3.4.2.1.4 <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.

3.4.2.1.5 <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.

3.4.2.1.6 <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.

3.4.2.1.7 <routes>

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

Container:
<scene-dump>

Attributes:
(none)

3.4.2.1.8 <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.

3.4.2.2 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>
```

3.4.3 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.

3.4.3.1 BIFS Feature List

The test suite shall verify the features in Table 3-4. 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 3.4.2. Rendering not being normative, the aspect of the node is not subject to conformance testing.

Table 3-4 — BIFS Test Suite Information

N°	Feature	Reference of Test sequence and associated method
1.	BIFS-Anim: position 3D animation	
2.	BIFS-Anim: position 2D animation	Anim-simple
3.	BIFS-Anim: color animation	
4.	BIFS-Anim: angle animation	Anim-circle
5.	BIFS-Anim: float animation	
6.	BIFS-Anim: bound float animation	
7.	BIFS-Anim: normal animation	
8.	BIFS-Anim: size 3D animation	
9.	BIFS-Anim: size 2D animation	
10.	BIFS-Anim: integer animation	
11.	BIFS-Anim: several fields in the same node	Anim-rect
12.	BIFS-Anim: several nodes	
13.	BIFS-Anim: skip frame	
14.	BIFS-Anim: switch of a node (isActive mask)	
15.	BIFS-Anim: random access true	
16.	BIFS-Anim: random access false	
17.	Quantization: 3D position	Pos3d-4bit
18.	Quantization: 2D position	
19.	Quantization: drawing order	
20.	Quantization: color	
21.	Quantization: texture coordinate	
22.	Quantization: angle	Angle-8bit
23.	Quantization: scale	
24.	Quantization: interpolator keys	
25.	Quantization: normals	Normal-4bit
26.	Quantization: rotations	
27.	Quantization: object size 3D	
28.	Quantization: object size 2D	
29.	Quantization: linear scalar quantization	
30.	Quantization: efficient float	
31.	Quantization: node default values	
32.	Quantization: isLocal mode	
33.	Quantization: DEF/USE	
34.	BIFS Command: insert node index	
35.	BIFS Command: insert node begin	
36.	BIFS Command: insert node end	Updatetest, Friday
37.	BIFS Command: insert Idx value index	
38.	BIFS Command: insert Idx value begin	
39.	BIFS Command: insert Idx value end	
40.	BIFS Command: insert ROUTE	
41.	BIFS Command: delete node	Bifs-deletenode
42.	BIFS Command: delete Idx value index	Friday
43.	BIFS Command: delete Idx value begin	
44.	BIFS Command: delete Idx value end	
45.	BIFS Command: replace node	
46.	BIFS Command: replace field	Bifs-2dfieldreplace1, Friday
47.	BIFS Command: replace Idx value index	Pae_raise
48.	BIFS Command: replace Idx value begin	
49.	BIFS Command: replace Idx value end	

N°	Feature	Reference of Test sequence and associated method
50.	BIFS Command: replace ROUTE	
51.	BIFS Command: replace scene	Ecran2, Updatetest
52.	BIFS Command: several commands in same AU	
53.	BIFS Scene: mask node	
54.	BIFS Scene: list node	Jerusalem, Layout, Testlayout
55.	BIFS Scene: mask MFField	
56.	BIFS Scene: list MFField	
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
63.	SFString	Ecran2, Updatetest
64.	SFTime	Jerusalem, OrientInterp3D
65.	SFUrl	Anchor, Audiotest
66.	SFVec2f	Ecran2, Updatetest
67.	SFVec3f	Bifs-deletenode
68.	SFImage	
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
74.	AnimationStream	Anim-rect, Anim-circle, Anim-simple
75.	Appearance	Bifs-deletenode, Bifs-2dfieldreplace1
76.	AudioBuffer	
77.	AudioClip	
78.	AudioDelay	
79.	AudioFX	
80.	AudioMix	
81.	AudioSource	Audiotest, Ifs
82.	AudioSwitch	
83.	Background	
84.	Background2D	
85.	Billboard	
86.	Bitmap	Ecran2, Jerusalem, Updatetest, Transition
87.	Box	Bifs-deletenode
88.	Circle	Bifs-2dfieldreplace1, Ecran2, Simple
89.	Collision	
90.	Color	
91.	ColorInterpolator	Timestest
92.	CompositeTexture2D	Layout
93.	CompositeTexture3D	
94.	Conditional	Ecran2, Layout, Friday
95.	Cone	Bifs-deletenode
96.	Coordinate	
97.	Coordinate2D	Layout, Updatetest
98.	CoordinateInterpolator	
99.	CoordinateInterpolator2D	

N°	Feature	Reference of Test sequence and associated method
100.	Curve2D	Layout, Polygontest
101.	Cylinder	Bifs-deletenode
102.	CylinderSensor	
103.	DirectionalLight	PointLightPrimitive1-3D
104.	DiscSensor	
105.	ElevationGrid	
106.	Expression	
107.	Extrusion	
108.	Face	
109.	FaceDefMesh	
110.	FaceDefTables	
111.	FaceDefTransform	
112.	FAP	
113.	FDP	
114.	FIT	
115.	Fog	Scaling3D
116.	FontStyle	Scaling3D, Ecran2
117.	Form	Form_spread, Form_spread2, Testform
118.	Group	Anchor, Ecran2, Layout
119.	ImageTexture	Ecran2, Jerusalem, Pae_raise
120.	IndexedFaceSet	Test2, Test3
121.	IndexedFaceSet2D	Ifs
122.	IndexedLineSet	
123.	IndexedLineSet2D	Polygontest, Updatetest
124.	Inline	
125.	LOD	
126.	Layer2D	Bifs-2dfieldreplace1, Transition
127.	Layer3D	Bifs-deletenode, Scaling3D
128.	Layout	Jerusalem, Layout, Testlayout
129.	LineProperties	Ecran2, Updatetest
130.	ListeningPoint	
131.	Material	Bifs-deletenode, Material3D
132.	Material2D	Bifs-2dfieldreplace1, Ecran2
133.	MovieTexture	Jerusalem, Friday
134.	NavigationInfo	
135.	Normal	
136.	NormalInterpolator	
137.	OrderedGroup	Form_spread2, Pae_raise
138.	OrientationInterpolator	OrientInterp3D
139.	PixelTexture	
140.	PlaneSensor	
141.	PlaneSensor2D	Slider, Valuator
142.	PointLight	PointLightPrimitive1-3D
143.	PointSet	
144.	PointSet2D	
145.	PositionInterpolator	Value_changed3d
146.	PositionInterpolator2D	Friday, Traj0
147.	ProximitySensor2D	
148.	ProximitySensor	
149.	QuantizationParameter	

N°	Feature	Reference of Test sequence and associated method
150.	Rectangle	Ecran2, Updatetest, Friday
151.	ScalarInterpolator	Trans-group
152.	Script	Scaling3D, SFColor01, Value_changed3d, Qtvr
153.	Shape	Bifs-deletenode, Ecran2, Jerusalem
154.	Sound	
155.	Sound2D	Audiotest, Ifs
156.	Sphere	Bifs-InsertNodeStress
157.	SphereSensor	
158.	SpotLight	PointLightPrimitive1-3D
159.	Switch	Ecran2, Jerusalem, Friday
160.	TermCap	
161.	Text	Ecran2, Jerusalem, Updatetest
162.	TextureCoordinate	
163.	TextureTransform	
164.	TimeSensor	OrientInterp3D, Jerusalem, Trans-group, Timestest
165.	TouchSensor	Scaling3D, Ecran2, Jerusalem, Friday
166.	Transform	Bifs-deletenode
167.	Transform2D	Bifs-2dfieldreplace1, Ecran2
168.	Valuator	Slider, Valuator
169.	Viewpoint	Scaling3D
170.	VisibilitySensor	
171.	Viseme	
172.	WorldInfo	
173.	DEF / USE	SFColor01, Ecran2, Jerusalem
174.	BIFSConfig (DecoderSpecificInfo for BIFS)	Ecran2, Jerusalem

3.4.3.2 OD Feature List

Table 3-5 — 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	

3.4.3.3 Bitstreams

Table 3-6

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
Updatetest	ENST	Updates
Transtion	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
Traj0	ENST	PositionInterpolator2D
Ifs	ENST	IndexedFaceSet2D and Sound2D
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
Test2	Optibase	IndexedFaceSet
Test3	Optibase	IndexedFaceSet

4 Visual

4.1 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.

4.2 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 of this part of ISO/IEC 14496 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).

4.2.1 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 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.

4.2.2 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.

4.2.3 Encoder requirements and recommendations

4.2.3.1 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.

4.2.3.2 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.

4.2.3.3 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.

4.3 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.

4.4 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. 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- and B-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 4.3.1). 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. 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 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. 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.

4.4.1 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).

4.4.2 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.

4.4.3 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.

4.4.4 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.

4.4.5 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).

4.4.6 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.

4.5 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.

4.5.1 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.

4.5.2 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.

4.5.3 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.

4.5.3.1 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.

4.5.3.1.1 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.

4.5.3.1.2 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.

4.5.3.1.3 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.

4.5.3.1.4 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.

4.5.3.1.5 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.

4.5.3.1.6 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.

4.5.3.1.7 Test bitstream #GE-9

This test has been removed from the test suite specification.

4.5.3.1.8 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.

4.5.3.1.9 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.

4.5.3.1.10 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.

4.5.3.1.11 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.

4.5.3.1.12 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.

4.5.3.1.13 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.

4.5.3.1.14 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.

4.5.3.1.15 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.

4.5.3.1.16 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).

4.5.3.1.17 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).

4.5.3.1.18 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].

4.5.3.1.19 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.

4.5.3.1.20 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).

4.5.3.1.21 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.

4.5.3.2 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 4-1

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:

4.5.3.2.1 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:

4.5.3.2.2 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.

4.5.3.2.3 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.

4.5.3.2.4 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.

4.5.3.2.5 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.

4.5.3.2.6 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:

4.5.3.2.7 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.

4.5.3.2.8 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.

4.5.3.2.9 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.

4.5.3.2.10 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.

4.5.3.2.11 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.

4.5.3.3 Test Bitstreams – Scalability

4.5.3.3.1 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)

4.5.3.3.2 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.

4.5.3.3.3 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.

4.5.3.3.4 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.

4.5.3.3.5 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.

4.5.3.3.6 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.

4.5.3.3.7 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

4.5.3.3.8 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.

4.5.3.3.9 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.

4.5.3.3.10 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

4.5.3.3.11 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

4.5.3.4 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

4.5.3.4.1 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.

4.5.3.4.2 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.

4.5.3.4.3 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.

4.5.3.4.4 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.

4.5.3.4.5 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.

4.5.3.4.6 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.

4.5.3.4.7 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.

4.5.3.4.8 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.

4.5.3.5 Test Bstreams - Error resilience

4.5.3.5.1.1 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.

4.5.3.5.1.2 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.

4.5.3.5.1.3 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.

4.5.3.6 Test Bitstreams - Scalable still texture

4.5.3.6.1 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 4-2

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 4-3

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.

4.5.3.7 Test Bitstreams - Sprites

4.5.3.7.1 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).

4.5.3.7.2 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).

4.5.3.7.3 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).

4.5.3.7.4 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).

4.5.3.7.5 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).

4.5.3.7.6 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).

4.5.4 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 4-1.

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.

```
[B]---->[S]---> (+) --[C]-------->  [O]
    |            ^
    |            |               Decoder under test
    |            |
    |          [MCP]<--[R]
    |                  [e]
    |                  [f]
    |                  [e]
    |                  [r]
    |                  [e]
    |                  [n]
    |                  [c]
    |          [MCP]<--[e]     Reference Decoder
    |            |
    |            |
    --->[S]---->(+) ---[C]--------> [O]

B:   test bitstream
S:   decoding processing units ISO/IEC 13818-2 subclauses 7.2 to 7.5
MCP: motion compensation unit (ISO/IEC 13818-2 subclause 7.6)
R:   reference frame
O:   output of decoder (reconstructed samples)
C:   clipping stage    [0,+255]
U:   current frame

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

Figure 4-1 — Frame buffer intercept method

4.5.5 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.

4.5.6 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.

4.5.7 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 4-4

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					

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-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												
	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											
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											
	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									

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-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											
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					

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

4.B.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 4-5 — 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 4-6 — 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)

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
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)
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

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
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 4-7 — 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 4-8 — 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	72	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 4-9 — 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

4.B.2 Core Profile bitstreams

Table 4-10 — 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 4-11 — 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
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 4-12 — 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 4-13 — 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 4-14 — 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 4-15 — 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 4-16 — 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 4-17 — 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 4-18 — 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 Audio

5.1 Introduction

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

Characteristics of coded bitstreams and decoders are defined for ISO/IEC 14496-3. 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 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 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 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 constructing tests to verify bitstream and decoder conformance. In addition, some test bitstreams implemented according to those guidelines are provided as the electronic attachment to this part of ISO/IEC 14496.

5.2 Audio Conformance Points

All audio decoders are part of the MPEG-4 System. The following interfaces have to be provided to test the audio decoders:

Table 5-1

DecoderSpecificInfo	The decoder specific information constitutes an opaque container with information for a specific media decoder.
Audio Access Units	See ISO/IEC 14496-1 subclause 7.2
Audio Composition Units	See ISO/IEC 14496-1 subclause 7.2
AudioSource Fields Information	See ISO/IEC 14496-1 subclause 7.2
Private Test Info	An interface to control some elements which are usually generated by random number generators.

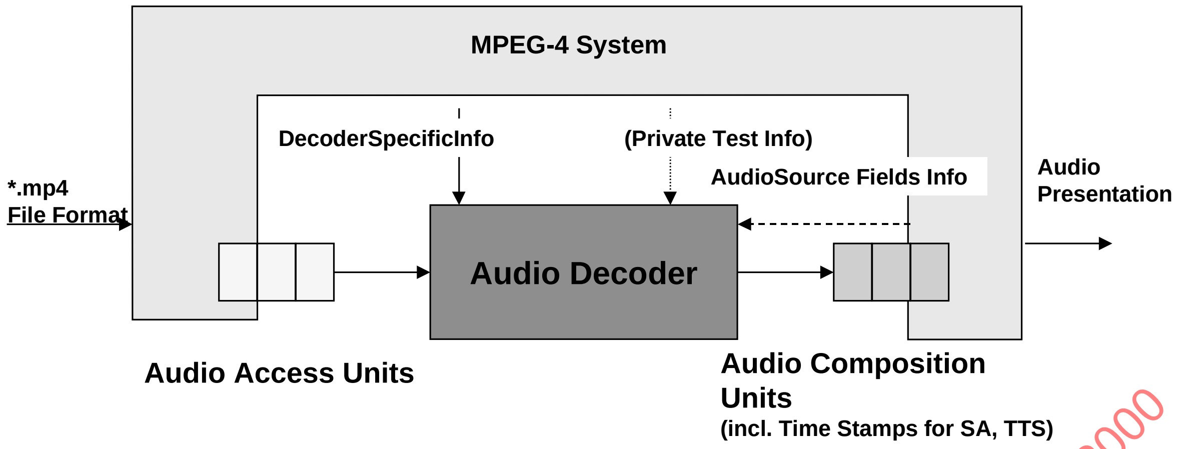

Figure 5-1 — Audio Conformance Points

5.3 Audio Profiles

Audio Profiles comprise a set of Audio Object Types. The conformance of a Profile is fulfilled, if the conformance of each Object Type, related to this Profile, is fulfilled. Four Audio Profiles have been defined:

1. The **Speech Audio Profile** provides a parametric speech coder, a CELP speech coder and a Text-To-Speech interface.

2. The **Synthesis Audio Profile** provides the capability to generate synthetic sound and speech at very low bitrates.

3. The **Scalable Audio Profile,** a superset of the Speech Profile, is suitable for scalable coding of speech and music, for transmission methods such as Audio on Internet and Digital Audio Broadcasting.

4. The **Main Audio Profile** is a rich superset of all the other Profiles: **Scalable Profile, Speech Profile, Synthesis Profile**, containing tools for natural and synthetic audio.

Table 5-2 — List of Audio Object Types

Audio Object Types	Speech Audio Profile	Synthesis Audio Profile	Scalable Audio Profile	Main Audio Profile
Null				
AAC LC			X	X
AAC main				X
AAC SSR				X
AAC LTP			X	X
AAC Scalable			X	X
TwinVQ			X	X
CELP	X		X	X
HVXC	X		X	X
TTSI	X		X	X
Main synthetic		X		X
Wavetable synthesis		(subset of Main synthetic)		(subset of Main synthetic)
General MIDI		(subset of Main synthetic)		(subset of Main synthetic)
Algorithmic Synthesis and Audio FX		(subset of Main synthetic)		(subset of Main synthetic)

5.4 Audio Interchange formats (Informative part)

5.4.1 Parsing an Audio_Data_Interchange_Format (ADIF) header

adif_id: shall be encoded with the value 0x41444946, the ASCII representation of the string “ADIF”.

5.4.2 Parsing Audio_Data_Transport_Stream (ADTS) header

5.4.2.1 adts_fixed_header

syncword: shall be encoded with the binary value 1111 1111 1111.

ID: shall not be encoded with the value 0.

layer: shall be encoded with the binary value 00.

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

sampling_frequency_index: shall be encoded with a value no greater than 0xb.

5.4.2.2 adts_variable_header

frame_length: shall be encoded with the length of the frame, including headers and error check (if present)

5.5 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.

5.5.1 General Object Type Descriptions

This chapter contains general descriptions of the bitstream and decoder conformance test. The descriptions may be related to one, more or all object types.

5.5.1.1 DecoderSpecificInfo Characteristics

DecoderSpecificInfo characteristics specify the constraints that are applied by the encoder in generating the DecoderSpecificInfo. These syntactic and semantic constraints may for example restrict the range or the values of parameters that are encoded directly or indirectly in the bitstream.

5.5.1.2 Audio Access Unit Characteristics

Audio Access Unit characteristics specify the constraints that are applied by the encoder in generating the Audio Access Unit. These syntactic and semantic constraints may for example restrict the range or the values of parameters that are encoded directly or indirectly in the Audio Access Unit.

5.5.1.3 AudioSource Fields Information Characteristics

AudioSource Fields Information Characteristics specify the constraints that exist in a MPEG-4 Player. AudioSource node fields like speed and pitch may change the behavior of the output of the Audio decoder.

5.5.1.4 Procedure to Test Bitstream Conformance

Each bitstream (DecoderSpecificInfo and Audio Access Units) 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 bitstreams. 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 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 into the appropriate levels.

5.5.1.5 Decoder Characteristics

A conforming decoder may support any of the free bitstream parameters in audio bitstreams. The decoder characteristics are defined by the profile and levels being tested.

5.5.1.6 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 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 the electronic attachment to this part of ISO/IEC 14496.

5.5.1.6.1 RMS Measurement

To be called an ISO/IEC 14496-3 audio decoder, the decoder shall provide an output 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^{-15}/sqrt(12). In addition, the difference signal shall have a maximum absolute value of at most 2^{-14} relative to full-scale.

For the calaculation 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 sine sweeps, with an amplitude of -20dB relative to full scale have an absolute amplitude of 0.1.

The test sequences 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 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.

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

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.

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

The decoder under test may be called an ISO/IEC 14496-3 audio decoder if the rms is less than 1/(2^15 * 12^0.5) and if the maximum absolute value of *diff* is less than or equal to 1/2^14.

5.5.1.6.2 Segmental SNR

This criterion is applied to the CELP, HVXC and TwinVQ object types. The definition to calculate the segmental SNR is given below.

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 decode 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

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

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

5.5.1.6.3 Frequency domain criterion based on cepstrum analysis

The cepstrum analysis procedure is defined by means of the functions lpc2cepstrum and calculate_lpc provided in pseude C code below. This criterion is used for testing conformance of TwinVQ and CELP decoders.

```
#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));
    }
}
```

5.5.1.6.4 Other Measurements Methods

For elements producing output that cannot be tested with the RMS method by direct comparison, for example AAC using PNS, HVXC and structured audio types (e.g. due to use of random signal generators), specific conformance testing procedures are described in the object type chapters itself.

5.5.1.7 Descriptions of the audio test bitstreams

All Test Bitstreams can be found at the publicly accessible FTP site (anonymous FTP) for MPEG4 conformance bitstreams.

The web site for information is:

http://www.research.att.com/projects/mpegaudio

The FTP site is:

ftp: **ftp.research.att.com**

user: **anonymous**

dir: **dist/mpegaudio**

File name convention:

For all test files, the file name convention given below is used.

Table 5-3 — test file name convention

Object Type Name	File Name
Null	NUxxx
AAC LC	ALxxx
AAC Main	AMxxx
AAC SSR	ASxxx
AAC LTP	APxxx
AAC scalable	ACxxx
Twin VQ	TVxxx
CELP	CExxx
HVXC	HVxxx
TTSI	TTSxx
Main Synthetic	MSxxx
Wavetable Synthesis	WSxxx
General MIDI	GMxxx
Algorithmic Synthesis and Audio FX	Syxxx
AudioBIFS	ABxxx
Audio Composition Unit Input for Composition Test	CUxxx

Content:

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

5.5.2 Null

The NULL object provides the possibility to feed raw PCM data directly into the audio compositor. No decoding is involved; however, an audio object descriptor is used to specify the sampling rate and the audio channel configuration.

5.5.3 Common Characteristics of the AAC-derived object types

The object types AAC LC (Low Complexity), AAC Main, AAC SSR (Scalable Sampling Rate) and AAC LTP (Long Term Prediction) build the basic objects 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, but uses a different decoder structure, syntax and additional tools to provide large step scalability.

The AAC LC, AAC Main and AAC SSR objects 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 decoder. The AAC Main and AAC LTP objects are built on top of the AAC LC object. The AAC SSR is identical to the AAC LC object 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 5-4 — AAC Object Types

Tools / Audio Object Type	13818-7 main	13818-7 LC	13818-7 SSR	PNS	LTP	TLSS	GA Bitstream Syntax Type	Hierarchy	Object Type ID
AAC main	X			X			ISO/IEC 13818-7 Style	contains AAC LC	1
AAC LC		X		X			ISO/IEC 13818-7 Style		2
AAC SSR			X	X			ISO/IEC 13818-7 Style		3
AAC LTP		X		X	X		ISO/IEC 13818-7 Style	contains AAC LC	4
AAC scalable		X		X	X	X	Scalable		5

The following characteristics apply to all 5 AAC-derived object types: AAC LC, AAC Main, AAC SSR, AAC LTP and AAC Scalable.

5.5.3.1 DecoderSpecificInfo Characteristics

There are several constraints for the values of DecoderSpecificInfo depending on the object type and therefore specified in the individual sections.

Within the limits of the profile and level, an encoder may apply restrictions to the following parameters of the DecoderSpecificInfo:

a) SamplingFrequencyIndex
b) SamplingFrequency (if SamplingFrequencyIndex = 0xf)
c) ChannelConfiguration
d) Program_config_element() (if applicable in the object type)

5.5.3.2 Audio Access Unit Characteristics

5.5.3.2.1 Decoding an individual_channel_stream

ics_reserved_bit: must be set to zero for all ObjectTypes except for AAC_scalable; for the latter ics_reserved_bit must be set to 1

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

5.5.3.2.2 Noiseless Coding

sect_cb[g][i]: shall not be encoded with the binary values 1100.

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.

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

hcod_sf[ ]: shall only be encoded with the values listed in the scalefactor Huffman table

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.

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

hcod_esc_y: shall be encoded with a value no larger than 8191, i.e., it shall be encoded with an initial escape sequence consisting of no more than nine ‘1’ bits followed by an escape separator of ‘0’.

hcod_esc_z: shall be encoded with a value no larger than 8191, i.e., it shall be encoded with an initial escape sequence consisting of no more than nine '1' bits followed by an escape separator of '0'.

5.5.3.2.3 Scalefactors

hcod_sf[ ]: shall only be encoded with the values listed in the scalefactor Huffman table.

5.5.3.2.4 Perceptual Noise Substitution

hcod_sf[ ]: shall only be encoded with the values listed in the scalefactor Huffman table.

5.5.3.2.5 Joint Coding

5.5.3.2.5.1 MS Stereo

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

5.5.3.2.5.2 Intensity Stereo

hcod_sf[ ]: shall only be encoded with the values listed in the scalefactor Huffman table.

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.

5.5.3.2.6 Coupling Channel

The number of dependently-switched and independently-switched coupling channel elements must not exceed the allowed numbers specified by the level and profile.

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.

5.5.3.2.7 Temporal Noise Shaping

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

5.5.3.2.8 IMDCT

window_sequence: The meaningful window_sequence transitions are as follows:

from ONLY_LONG_SEQUENCE to $\left\{\begin{array}{l}\text{ONLY\_LONG\_SEQUENCE}\\ \text{LONG\_START\_SEQUENCE}\end{array}\right.$

from LONG_START_SEQUENCE to $\left\{\begin{array}{l}\text{EIGHT\_SHORT\_SEQUENCE}\\ \text{LONG\_STOP\_SEQUENCE}\end{array}\right.$

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 must consist of only meaningful window_sequence transitions. However, decoders are required to handle non-meaningful window_sequence transitions as well. Test bitstreams AL3 and AS17 are provided respectively for Main and Low-Complexity profiles to test decoder performance on non-meaningful transitions (see subclause 5.5.5.3.2.2). The performance requirements for non-meaningful window_sequence transitions are the same as for the meaningful transitions.

5.5.3.2.9 LFE

The number of LFEs must 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.

The highest non-zero spectral coefficient of any LFE shall be 12.

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.

5.5.3.2.10 Program Configuration Element

Program Configuration Elements in Access Units shall be ignored. Therefore, PCEs transmitted in Access Units cannot be used to convey decoder configuration information. The PCE in the DecoderSpecificInfo describes the decoder information for the elementary stream under consideration.

5.5.3.3 Decoder Characteristics

A conformant decoder shall support all characteristics given by the profile and level. Thus only bitstreams belonging to the specific level & profile have to be tested.

5.5.3.4 Procedure to Test Decoder Conformance

The test procedure ‘Calculation of RMS’ applies to all sine sweep signals. This test only verifies the computational accuracy of an implementation. For all test bitstreams containing PNS data, an energy analysis in time and/or frequency domain is required (see subclause 5.5.3.4.2). 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. This also applies to bitstreams with non-meaningful window sequences.

5.5.3.4.1 Calculation of RMS

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 the electronic attachment to this part of ISO/IEC 14496 using the procedure described in subclause 5.5.1.6.1. 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. To be called an ISO/IEC 14496-3 audio decoder, the decoder shall provide an output 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^{-15}/sqrt(12). In addition, the difference signal shall have a maximum absolute value of at most 2^{-14} relative to full-scale.

This test only verifies the computational accuracy of an implementation.

5.5.3.4.2 Calculation of PNS conformance criteria

In order to test the PNS tool, special bitstreams only containing PNS data are provided. Two criteria can then be applied. A test method based on spectral (PNS-S) and temporal (PNS-T) analysis of the decoded signal with respect to the reference signal.

Spectral PNS conformance criterium (PNS-S)

1. A bitstream (AL09) is supplied containing a static spectrum generated by PNS and Null codebook sections (i.e. each frame carries the same spectral "envelope", long blocks only, no other codebooks). Both the decoded output and the reference output signal are analyzed by means of an 2048-point DFT with a Hann window and 50% overlap between subsequent windows. For both signals, the DFT lines are grouped corresponding to scalefactor bands (see Table 8.4 ISO/IEC 13818-7) and the accumulated squared absolute values are computed for each scalefactor band. As the first conformance criterion, the ratio between the energies of both signals averaged over time must be within the interval [-0.4 dB;0.4 dB] for each scalefactor band. As the second conformance criterion, the ratio between the standard deviations (overtime) of the energies of both signals must be within the interval [-4.0 dB;1.0 dB] for each scalefactor band.

2. A bitstream (AL10) is 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 0,2,4,5). The same type of analysis is used as in condition (1), but with a window size of 256 and grouping corresponding to scalefactor bands for a SHORT_WINDOW (see Table 8.5 ISO/IEC 13818-7). For the first criterion the ratio now must be within the interval [-0.4 dB;0.4 dB], for the second within the interval [-4.0 dB;2.0 dB].

Temporal PNS conformance criterium (PNS-T)

3. For bitstream (AL11), an additional conformance criterion is used: 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 a conformance criterion, the ratio between the energies of both signals must be within the interval [-5 dB;5 dB] for 85% of the blocks and within the interval [-10 dB;10 dB] for 98% of the blocks.

5.5.4 Common characteristics of the AAC objects supporting ISO/IEC 13818-7 profiles (AAC LC, AAC Main, AAC SSR and AAC LTP)

These four object types support the Syntax of ISO/IEC 13818-7. The following characteristics apply to all of them.

5.5.4.1 DecoderSpecificInfo Characteristics

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 0

Program_config_element(): Any program may contain no more main audio channels, LFE channels, independent coupling_channel_elements, and dependent coupling_channel_elements 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: no restrictions

sampling_frequency_index: shall match the samplingFrequencyIndex within AudioSpecificConfig

mono_mixdown_element_number: must be encoded with the element_instance_tag of a single_channel_element.

stereo_mixdown_element_number: must 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.

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

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

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

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

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

cc_element_is_ind_sw: must 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: must be encoded with the element_instance_tag of a coupling_channel_element.

Within the limits of the profile and level, an encoder may apply restrictions to the following parameters of the DecoderSpecificInfo:

a) SamplingFrequencyIndex
b) SamplingFrequency
c) ChannelConfiguration
d) Program_config_element()

5.5.4.2 Audio Access Unit Characteristics

5.5.4.2.1 Decoding of raw data blocks

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

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_elements (DSE), which may have duplicated element_instance_tags.

5.5.4.2.2 Noiseless Coding

pulse_data_present: shall be encoded with a value of 0 when window_sequence is EIGHT_SHORT_SEQUENCE.

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] must be no greater than 1023.

pulse_amp[i]: shall be encoded with a value small enough such that the compensated quantized spectral coefficient is no greater than 8191.

5.5.4.3 Decoder Characteristics

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

Table 5-5 — AAC Parameter

Bitstream Characteristic	Variation
data_stream_element	a decoder is not required to store or present data recovered from data_stream_elements
mono-mixdown element	a decoder is not required to present audio from the mono-mixdown element
stereo-mixdown element	a decoder is not required to present audio from the stereo-mixdown element
matrix-mixdown	a decoder is not required to calculate a matrix-mixdown signal

5.5.5 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 a 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.

5.5.5.1 DecoderSpecificInfo Characteristics

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

a) SamplingFrequencyIndex
b) SamplingFrequency
c) ChannelConfiguration
d) sampling_frequency_index
e) mono_mixdown_element
f) stereo_mixdown_element
g) pulse_data
h) window_shape
i) program_config_element
j) M/S stereo
k) intensity stereo
l) TNS
m) data_stream_element

n) dependently switched coupling channel
o) independently switched coupling channel
p) LFE channel
q) matrix-downmix

5.5.5.2 Audio Access Unit Characteristics

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:

a) sampling frequency_index
b) mono_mixdown_element
c) stereo_mixdown_element
d) use of prediction in main profile
e) pulse_data
f) window_shape
g) program_config_element
h) M/S stereo
i) intensity stereo
j) TNS
k) data_stream_element
l) dependently switched coupling channel
m) independently switched coupling channel
n) LFE channel
o) matrix-downmix

5.5.5.3 Procedure to Test Bitstream Conformance

5.5.5.3.1 DecoderSpecificInfo

AudioObjectType: Shall be encoded with the value 2

SamplingFrequencyIndex: Shall be encoded with the following values:

Table 5-6

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

SamplingFrequency: Shall be encoded with the following values:

Table 5-7

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

ChannelConfiguration: Shall be encoded with the following values:

Table 5-8

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

The following restrictions apply to GASpecificConfig:

frameLength: 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 0

program_config_element(): A program cannot contain more main audio channels, LFE channels, independent coupling_channel_elements, and dependent coupling_channel_elements than is specified by the profile and level.

object_type: shall be encoded with the value 1

sampling_frequency_index: shall match the samplingFrequencyIndex within AudioSpecificConfig

mono_mixdown_element_number: must be encoded with the element_instance_tag of a single_channel_element.

stereo_mixdown_element_number: must 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.

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

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

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

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

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

cc_element_is_ind_sw: must 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: must be encoded with the element_instance_tag of a coupling_channel_element.

5.5.5.3.2 Audio Access Units

5.5.5.3.2.1 raw_data_block()

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

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_elements (DSE), which may have duplicated element_instance_tags.

5.5.5.3.2.2 ics_info() and individual_channel_stream()

ics_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 conformant bitstream 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

predictor_data_present: shall be encoded with the value 0

gain_control_data_present: shall be encoded with the value 0

5.5.5.3.2.3 Noiseless Coding

sect_cb[g][i]: shall not be encoded with the binary values 1100.

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.

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

hcod_sf[ ]: shall only be encoded with the values listed in the scalefactor Huffman table, Table A-1.

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.

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

hcod_esc_y: shall be encoded with a value no larger than 8191, i.e., it shall be encoded with an initial escape sequence consisting of no more than nine ‘1’ bits followed by an escape separator of ‘0’.

hcod_esc_z: shall be encoded with a value no larger than 8191, i.e., it shall be encoded with an initial escape sequence consisting of no more than nine ‘1’ bits followed by an escape separator of ‘0’.

pulse_data_present: shall be encoded with a value of 0 when window_sequence is EIGHT_SHORT_SEQUENCE.

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] must be no greater than 1023.

pulse_amp[i]: shall be encoded with a value small enough such that the compensated quantized spectral coefficient is no greater than 8191.

5.5.5.3.2.4 Scalefactors

hcod_sf[ ]: shall only be encoded with the values listed in the scalefactor Huffman table, Table A-1.

5.5.5.3.2.5 Joint Coding

5.5.5.3.2.5.1 MS Stereo

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

5.5.5.3.2.5.2 Intensity Stereo

hcod_sf[ ]: shall only be encoded with the values listed in the scalefactor Huffman table, Table A-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.

5.5.5.3.2.6 Coupling Channel

The number of dependently-switched and independently-switched coupling channel elements must not exceed the allowed numbers specified by the profile and level.

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 profile and level.

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.

5.5.5.3.2.7 Temporal Noise Shaping

length[w][filt]: must 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]: must not exceed the maximum permitted order depending on the specified profile and level

5.5.5.3.2.8 LFE

The number of LFEs must 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. The highest non-zero spectral coefficient of any LFE shall be 12. 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.

5.5.5.3.2.9 Program Configuration Elements

A program cannot contain more main audio channels, LFE channels, independent coupling_channel_elements, and dependent coupling_channel_elements than is specified by the profile and level.

mono_mixdown_element_number: must be encoded with the element_instance_tag of a single_channel_element.

stereo_mixdown_element_number: must 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.

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

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

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

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

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

cc_element_is_ind_sw: must 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: must be encoded with the element_instance_tag of a coupling_channel_element.

5.5.5.4 Decoder Characteristics

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

Table 5-9 — AAC Parameter

Bitstream 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
program configuration	a decoder is only required to decode one program of a multi-program bitstream
data_stream_element	a decoder is not required to store or present data recovered from data_stream_elements
mono mixdown	a decoder is not required to present audio from the mono-mixdown element
stereo mixdown	a decoder is not required to present audio from the stereo-mixdown element
matrix mixdown	a decoder is not required to calculate a matrix-mixdown signal

5.5.5.5 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. These test sequences are provided for sampling rates of 8, 11.025, 12, 16, 22.05, 24, 32, 44.1, 48, 64, 88.2, and 96 kHz. The test set includes a sine sweep and musical test sequences, as listed in Table 5-10. The extension _fs is appended to the bitstream name to indicate the sampling rate of the test sequence. Possible values of fs are 8, 11, 12, 16, 22, 24, 32, 44, 48, 64, 88, and 96, corresponding to the possibly non-integer sampling rates listed above. For each bitstream, two bitrates are listed in the table. 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.

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 the electronic attachment to this part of ISO/IEC 14496 using the procedure described in subclause 5.5.1.6.1. 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. To be called an ISO/IEC 14496-3 audio decoder, the decoder shall provide an output 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^{-15}/sqrt(12). In addition, the difference signal shall have a maximum absolute value of at most 2^{-14} relative to full-scale. This test only verifies the computational accuracy of an implementation.

For all test bitstreams containing PNS data, an energy analysis in time and/or frequency domain is required (see subclause 5.5.3.4.2).

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. This also applies to bitstreams with non-meaningful window sequences.

5.5.5.5.1 Perceptual Noise Substitution (PNS)

See subclause 5.5.3.4.2.

5.5.5.5.2 Calculation of RMS

See subclause 5.5.1.6.1.

5.5.5.6 Descriptions of Conformance Bitstreams

All bitstreams will be supplied in all sampling rates (as indicated by extension _fs). For a specific profile and level only the bitstreams with the appropriate channel configuration and sampling rate are applicable.

Table 5-10 — AAC-LC Object Type Test Bitstreams

File Name	AL00	AL01	AL02	AL03	AL04	AL05	AL06	AL07	AL08
Supplier	AT&T	AT&T	AT&T	AT&T	AT&T	AT&T	AT&T	AT&T	AT&T
Content	sine sweep	music	music	music	music	music	music	music, other	music
Bitrate	40/64	40/64	40/64	40/64	40/64	80/128	120/ 192	240/ 384	1920/ 3072
# single channel elements	1	1	1	1	1	0	1	1	16
# channel pair elements	0	0	0	0	0	1	1	2	16
# LFE channels	0	0	0	0	0	0	0	1	0
# Dep coupling channels	0	0	0	0	0	0	0	1	0
Data stream elements		Yes							
Intensity						Yes	Yes	Yes	Yes
MS						Yes	Yes	Yes	Yes
Prediction									
Window shape					Yes		Yes	Yes	Yes
TNS					Yes		Yes	Yes	Yes
Pulse data					Yes		Yes	Yes	Yes
Buffer test			Yes						
Nonmeaningful window_sequence transitions				Yes					
LTP									
PNS									
Test procedure	RMS								

Table 5-10 — AAC-LC Object Type Test Bitstreams (continued)

File Name	AL09	AL10	AL11
Supplier	Fhg	Fhg	Fhg
Content	see subclause 5.5.3.4.2	see subclause 5.5.3.4.2	see subclause 5.5.3.4.2
Bitrate	40/64	40/64	40/64
# single channel elements	1	1	1
PNS	Yes	Yes	Yes

5.5.6 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 a 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. All AAC LC Object Type conformance points are also necessary for this Object Type.

5.5.6.1 DecoderSpecificInfo Characteristics

These characteristics specify the constraints that are applied by the encoder in generating the DecoderSpecificInfo. In addition to restrictions specified for the AAC-LC object type, encoders may apply restrictions to the following parameters of the DecoderSpecificInfo:

a) use of prediction

5.5.6.2 Audio Access Unit Characteristics

These characteristics specify the constraints that are applied by the encoder in generating the Audio Access Units. In addition to restrictions specified for the AAC-LC object type, encoders may apply restrictions to the following parameters of the Audio Access Units:

a) use of prediction

5.5.6.3 Procedure to Test Bitstream Conformance

5.5.6.3.1 DecoderSpecificInfo

AudioObjectType: Shall be encoded with the value 1

SamplingFrequencyIndex: Shall be encoded with values from 0 to 0xc

SamplingFrequency: Shall have values from 0 to 96000

ChannelConfiguration: Shall be encoded with values from 0 to 7

The restrictions on GASpecificConfig parameters specified for AAC LC object type also apply for AAC Main object type.

5.5.6.3.2 Audio Access Units

All restrictions for AAC LC object type apply except as specified below.

5.5.6.3.2.1 Prediction

predictor_data_present: no restrictions apply

predictor_reset: shall be encoded with the binary value of 1 for at least one out of every eight consecutive frames for programs in which predictor_data_present is 1.

predictor_reset_group_number: shall not be encoded with the binary values 00000 or 11111; every valid value shall occur once within a consecutive series of 30 occurrences of pred_reset_group_number.

5.5.6.3.2.2 Temporal Noise Shaping

order[w][filt]: must not exceed the maximum permitted order depending on the specified profile and level.

5.5.6.4 Decoder Characteristics

All restrictions for AAC LC object type apply.

5.5.6.5 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. Sequences are provided for sampling rates of 8, 11.025, 12, 16, 22.05, 24, 32, 44.1, 48, 64, 88.2, and 96 kHz. The test set includes a sine sweep and musical test sequences, as listed in Table 5-11. The extension _fs is appended to the bitstream name to indicate the sampling rate of the test sequence. Possible values of fs are 8, 11, 12, 16, 22, 24, 32, 44, 48, 64, 88, and 96, corresponding to the possibly non-integer sampling rates listed above. For each bitstream, two bitrates are listed in the table. 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.

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 the electronic attachment to this part of ISO/IEC 14496 using the procedure described in the subclause "RMS Measurement." 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. To be called an ISO/IEC 14496-3 audio decoder, the decoder shall provide an output 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^{-15}/sqrt(12). In addition, the difference signal shall have a maximum absolute value of at most 2^{-14} relative to full-scale. 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. This also applies to bitstreams with non-meaningful window sequences.

5.5.6.6 Descriptions of Conformance Bitstreams

All bitstreams will be supplied in all sampling rates (carrying an additional index _fs).

For a specific profile only the bitstreams with the appropriate channel configuration and sampling rate are applicable. A AAC Main object type decoders must also be able to decode all AAC LC Object Type bitstreams.

Table 5-11 — AAC Main Object Type Test Bitstreams

File Name	AM00	AM01	AM02	AM03
Supplier	AT&T	AT&T	AT&T	AT&T
Content	sine sweep	music	music	music
Bitrate	40/64	40/64	80/128	240/512
# single channel elements	1	1	0	1
# channel pair elements	0	0	1	2
# LFE channels	0	0	0	1
# Dep coupling channels	0	0	0	1
# Indep Coupling Channels	0	0	0	2
Intensity			Yes	Yes
MS			Yes	Yes
Prediction	Yes	Yes	Yes	Yes
Window shape			Yes	Yes
TNS			Yes	Yes
Pulse data				
Buffer test				
Nonmeaningful window_sequence transitions				
LTP				
PNS				

5.5.7 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.

5.5.7.1 DecoderSpecificInfo Characteristics

These characteristics specify the constraints that are applied by the encoder in generating the DecoderSpecificInfo. All restrictions specified for the AAC-LC object type apply.

5.5.7.2 Audio Access Unit Characteristics

These characteristics specify the constraints that are applied by the encoder in generating the Audio Access Units All restrictions specified for the AAC-LC object type apply.

5.5.7.3 Procedure to Test Bitstream Conformance

5.5.7.3.1 DecoderSpecificInfo

AudioObjectType: Shall be encoded with the value 3

SamplingFrequencyIndex: Shall be encoded with values from 0 to 0xc

SamplingFrequency: Shall have values from 0 to 96000

ChannelConfiguration: Shall be encoded with values from 0 to 7

The restrictions on GASpecificConfig parameters specified for AAC LC object type also apply for AAC SSR object type.

5.5.7.3.2 Audio Access Units

All restrictions for AAC LC object type apply except as specified below.

5.5.7.3.2.1 Temporal Noise Shaping

order[w][filt]: must not exceed the maximum permitted order depending on the specified profile and level.

5.5.7.3.2.2 Gain Control

aloccode: must satisfy the following conditions:

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

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

5.5.7.4 Decoder Characteristics

All restrictions for AAC LC object type apply.

5.5.7.5 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. Sequences are provided for sampling rates of 8, 11.025, 12, 16, 22.05, 24, 32, 44.1, 48, 64, 88.2, and 96 kHz. The test set includes a sine sweep and musical test sequences, as listed in Table 5-12. The extension _fs is appended to the bitstream name to indicate the sampling rate of the test sequence. Possible values of fs are 8, 11, 12, 16, 22, 24, 32, 44, 48, 64, 88, and 96, corresponding to the possibly non-integer sampling rates listed above. For each bitstream, two bitrates are listed in the table. 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.

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 the electronic attachment to this part of ISO/IEC 14496 using the procedure described in subclause 5.5.1.6.1. 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.To be called an ISO/IEC 14496-3 audio decoder, the decoder shall provide an output 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^{-15}/sqrt(12). In addition, the difference signal shall have a maximum absolute value of at most 2^{-14} relative to full-scale. 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. This also applies to bitstreams with non-meaningful window sequences.

5.5.7.6 Descriptions of Conformance Bitstreams

All bitstreams will be supplied in all sampling rates (carrying an additional index _fs). For a specific profile only the bitstreams with the appropriate channel configuration and sampling rate are applicable.

Table 5-12 — AAC SSR Object TypeTest Bitstreams

File Name	AS00	AS01	AS02	AS03	AS04	AS05	AS06	AS07	AS08
Supplier	Sony	Sony	Sony	Sony	Sony	Sony	Sony	Sony	Sony
Content	sine sweep	music	music	music	music	music	music	music	music
Bitrate	40/64	40/64	40/64	40/64	40/64	40/64	40/64	40/64	40/64
# single channel elements	1	1	1	1	1	1	1	1	1
# channel pair elements	0	0	0	0	0	0	0	0	0
# LFE channels	0	0	0	0	0	0	0	0	0
Data stream elements						Yes	Yes	Yes	Yes
Intensity									
Prediction									
MS									
Gain Control		Yes	Yes	Yes					
Window shape		Yes	Yes	Yes	Yes				
TNS		Yes	Yes	Yes	Yes				
pulse data		Yes	Yes	Yes	Yes				
Buffer test		Yes	Yes	Yes	Yes				
Bandwidth		4	3	2	1	4	3	2	1
Nonmeaningful window_sequence transitions									

Table 5-12 — AAC SSR Object TypeTest Bitstreams (continued)

File Name	AS09	AS10	AS11	AS12	AS13	AS14	AS15	AS16	AS17
Supplier	Sony	Sony	Sony	Sony	Sony	Sony	Sony	Sony	Sony
Content	music	music	music	music	music	music	music	music	music
Bitrate	80/ 128	80/ 128	80/ 128	80/ 128	200/ 320	200/ 320	280/ 448	280/ 448	40/64
# single channel elements	0	0	0	0	1	1	1	1	1
# channel pair elements	1	1	1	1	2	2	3	3	0
# LFE channels	0	0	0	0	1	1	1	1	0
Data stream elements									
Intensity	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	
Prediction									
MS	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	
Gain Control	Yes	Yes	Yes		Yes	Yes	Yes	Yes	Yes
Window shape					Yes	Yes	Yes	Yes	
TNS					Yes	Yes	Yes	Yes	
pulse data					Yes	Yes	Yes	Yes	
Buffer test									
Bandwidth	4	3	2	1	4	3	4	3	4
Nonmeaningful window_sequence transitions									Yes

5.5.8 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, a 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. A MPEG-2 AAC LC profile bitstream can be decoded without restrictions.

5.5.8.1 Decoder Specific Config Characteristics

AudioObjectType: Shall be encoded with the value 4

SamplingFrequencyIndex: Shall be encoded with the following values:

Table 5-13

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

SamplingFrequency: Shall be encoded with the following values:

Table 5-14

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

ChannelConfiguration: Shall be encoded with the following values:

Table 5-15

ChannelConfiguration	Level 1	Level 2	Level 3	Level 4
Scalable Profile	1	1..2	1..2	1..2
Main Profile	1..2			

5.5.8.2 Decoder Characteristics

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

5.5.8.3 Descriptions of the audio test bitstreams

Table 5-16 — AAC LTP Object Type Test Bitstreams

File Name	AP01	AP02	AP03	AP04	AP05
Supplier	Nokia	Nokia	Nokia	Nokia	Nokia
Content	sine sweep	music	music	music	music
Bitrate	64	128	64	64	128
Sampling rate	48	48	48	48	48
# single channel elements	1		1	1	
# channel pair elements		1			1
# Indep Coupling Channels					
Intensity					Yes
MS		Yes			Yes
Tns		Yes	Yes		Yes
Long-term prediction	Yes	Yes	Yes	Yes	Yes

5.5.9 AAC scalable

The scalable AAC object is built on top of the AAC LTP object, 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.

5.5.9.1 Decoder Specific Config Characteristics

The following restrictions apply to AudioSpecificConfig:

AudioObjectType: Shall be encoded with the value 6

SamplingFrequencyIndex: Shall be encoded with the following values:

Table 5-17

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

SamplingFrequency: Shall be encoded with the following values:

Table 5-18

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

ChannelConfiguration: Shall be encoded with the following values:

Table 5-19

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

The following restrictions apply to GASpecificConfig:

FrameLength: no restrictions

DependsOnCoreCoder: no restrictions

CoreCoderDelay: no restrictions

ExtensionFlag: shall be encoded with the value 0

5.5.9.2 Audio Access Unit Characteristics

5.5.9.2.1 Noiseless Coding

pulse_data_present: shall be encoded with a value of 0.

5.5.9.3 Descriptions of the audio test bitstreams

All GA Main scaleable profile decoders must also be able to decode all GA LC scaleable objects.

5.5.10 AAC-based Scalable Configurations

5.5.10.1 Object combination restrictions

puls data: obsolet

A decoder shall support all object type combinations specified in ISOIEC 14496-3.

5.5.10.2 Sampling rate and upsampling filter restrictions

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.

5.5.10.3 Description of the audio test bitstreams

Table 5-20 — AAC Scalable Object Test Bitstreams

File Name	AC01	AC02	AC03	AC04	AC05	AC06
mnemonic	LTP-M	LTP M-S	LTP S	CELP M-S	CELP M-S2	CELP S
supplier	FhG	FhG	FhG	FhG	FhG	FhG
content						
bitrate [kbit/s]	32+7*16	32+3*16+4*32	64+7*32	X+7*32	X+Y+3*16+4*32	X+Y+32+3*32
sampling rate [kHz]	48	48	48	7.35/44.1	8/48	7.35/22.05
frame length	1024	1024	1024	960	960	960
LTP	yes	yes	yes	-	-	-
CELP	-	-	-	1	2	2
TwinVQ (mono)	-	-	-	-	-	-
TwinVQ (stereo)	-	-	-	-	-	-
AAC base (mono)	1	1	-	-	-	
AAC base (stereo)	-	-	1	-	-	1
AAC extension (mono)	7	3	-	3	3	-
Core Coder Delay				8000	0	
AAC extension (stereo)	-	4	7	4	4	3

Table 5-20 — AAC Scalable Object Test Bitstreams (continued)

File Name	AC07	AC08	AC09	AC10	AC11	AC12	AC13
mnemonic	TVQ S AAC S	LTP/TVQ M AAC M	TVQ M AAC M S	TVQ M AAC S	LTP/TVQ S AAC S	CELP M-S	CELP M-S
supplier	FhG	FhG	FhG	FhG	FhG	FhG	FhG
content							
bitrate [kbit/s]	2*16+32+ 7*32	2*16+16+ 7*16	2*16+16+ 3*16+4*32	2*16+32+ 7*32	2*24+32+ 7*32	6+16+32	6+32+64
sampling rate [kHz]	96	48	48	22.05	44.1	8/8	12/24
frame length	1024	1024	1024	1024	1024	960	960
LTP	-	yes	-	-	-	-	-
CELP	-	-	-	-	-	1	1
TwinVQ (mono)	-	2	2	2	-	-	-
TwinVQ (stereo)	2	-	-	-	2	-	-
AAC base (mono)	-	1	1	-	-	1	1
AAC base (stereo)	1	-	-	1	1	-	-
AAC extension (mono)	-	7	3	-	-	-	-
TNS base layer in first AAC layer	x	x	x	x	x	x	x
TNS ext layer			x				
MS-Stereo	x		x	x	x		
AAC extension (stereo)	7	-	4	3	7	-	-

Table 5-20 — AAC Scalable Object Test Bitstreams (continued)

File Name	AC14	AC15	AC16	AC17	AC18	AC19
mnemonic	CELP M-S	CELP M-S	TVQ M AAC M-S	TVQ M AAC M-S	TVQ M AAC M-S	TVQ M AAC M-S
supplier	FhG	FhG	FhG	FhG	FhG	FhG
content						
bitrate [kbit/s]	6+64+128	6+64+128	8+16+32	16+32+64	16+64+128	32+96+192
sampling rate [kHz]	8/32	8/96	12	24	32	64
frame length	960	960	1024	1024	1024	1024
LTP	-	-	yes	yes	yes	yes
CELP	1	1	-	-	-	-
TwinVQ (mono)	-	-	1	1	1	1
TwinVQ (stereo)	-	-	-	-	-	-
AAC base (mono)	-	1	1	1	1	1
AAC base (stereo)	1	-	-	-	-	-
AAC extension (mono)	-	-	-	-	-	-
AAC extension (stereo)	1	1	1	1	1	1

Table 5-20 — AAC Scalable Object Test Bitstreams (continued)

File Name	AC20	AC21	AC22
mnemonic	AAC Tools1	AAC-Tools3	AAC-Tools4
supplier	FhG	FhG	FhG
content			
bitrate [kbit/s]	6+4*32	4*32	4*32
sampling rate [kHz]	48	48	48
frame length	960	1024	1024
LTP	-	-	yes
CELP	-	-	-
TwinVQ	yes	-	-
TNS (TwinVQ)	yes	-	-
TNS (AAC base)	-	-	yes
TNS (AAC extension)	yes	-	yes
Intensity layer	-	-	1,2
MS stereo	yes	yes	yes
PNS in layer	-	1,2,3,4	-

5.5.11 TwinVQ

5.5.11.1 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).

5.5.11.2 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)

5.5.11.3 Procedure to Test Bitstream Conformance

5.5.11.3.1 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.

5.5.11.3.2 Decoding of the payload

5.5.11.3.2.1 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.

5.5.11.3.2.2 parsing tvq_scalable_extension_header()

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

5.5.11.3.2.3 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.

5.5.11.4 Decoder Characteristics

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

5.5.11.5 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 5-21. 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 decoders

The Conformance criteria for Full Accuracy TwinVQ decoder are based on a maximum absolute difference signal value and a maximum RMS value over the full length of the difference signal.
To be called an ISO/IEC 14496-3 Full Accuracy TwinVQ decoder, the decoder shall provide an output 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 $\frac{2^{-15}}{\sqrt{12}}$. In addition, the difference signal shall have a maximum absolute value of at most 2^{-14} relative to full scale.

Conformance criteria for Fixed-Point Accuracy 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.

5.5.11.6 Descriptions of the audio test bitstreams

Table 5-21 — TwinVQ Object Type Test Bitstreams

File Name	TV00	TV01	TV02	TV03	TV04	TV05	TV06	TV07
supplier **-**	NTT	NTT	NTT	NTT	NTT	NTT	NTT	NTT
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 5-21 — TwinVQ Object Type Test Bitstreams (continued)

File Name	TV10	TV11	TV12	TV13	TV14	TV15	TV16
supplier **-**	NTT	NTT	NTT	NTT	NTT	Matsushita	Matsushita
content	music	music	music	music	music	music	music
level in scalable profile	2	2	1	1	1	3	2
bitrate [kbit/s]	16	16 + 16	8	8+8	8+8+8	32+32+32	12*8
sampling rate [kHz]	16	16	24	24	24	48	44.1
frame length	1024	1024	1024	1024	1024	1024	1024
number of scaleable layers	1	1	1	2	3	3	8
number of channels	2	2	1	1	1	2	1
long-term prediction (LTP)	no	no	no	no	no	no	no
adaptive window shape	no	no	no	no	no	no	no
TNS	no	no	no	no	no	no	no
M/S stereo mode 2	yes	yes	-	-	-	yes	-
M/S stereo mode 1	yes	yes	-	-	-	no	-
bandlimit_present	no	no	no	no	no	no	no
ppc_present	no	no	no	no	no	no	no
postprocess_present	no	no	no	no	no	no	no

Table 5-21 — TwinVQ Object Type Test Bitstreams (continued)

File Name	TV20	TV21	TV22	TV23	TV24	TV25	TV26	TV27
supplier	IPA	IPA	IPA	IPA	IPA	IPA	IPA	IPA
content	music	music	music	music	music	music	music	music
level in scalable profile	1	2	3	1	1	2	3	1
bitrate [kbit/s]	8	16	32	16	16+16	16+16+16	32+32	16+16+16
sampling rate [kHz]	16	44.1	48	24	16	32	32	22.05
frame length	1024	1024	1024	1024	1024	1024	1024	1024
number of scaleable layers	1	1	1	1	2	3	2	3
number of channels	1	1	2	1	1	1	2	1
long-term prediction (LTP)	no	no	no	yes	no	no	no	yes
adaptive window shape	no	yes	no	no	no	yes	no	no
TNS	no	no	no	yes	no	no	no	yes
M/S stereo mode 2	-	-	yes	-	-	-	yes	-
M/S stereo mode 1	-	-	yes	-	-	-	yes	-
bandlimit_present	no	yes	no	no	no	yes	no	no
ppc_present	no	yes	no	no	no	yes	no	no
postprocess_present	no	yes	no	no	no	yes	no	no
scan all codebook	yes	no	no	no	yes	no	no	no

5.5.12 CELP

5.5.12.1 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

5.5.12.2 Audio Access Unit characteristics

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

a) LPC_Present
b) interpolation_flag
c) gain_indices [1]

5.5.12.3 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 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 objects.

channelConfiguration: must be set to 1.

AudioSpecificConfig – Speech Profile

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

AudioObjectType: must be set to 8 for CELP objects.

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 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 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 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.

5.5.12.4 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 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 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 objects 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.

5.5.12.5 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 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 criteria for Full Accuracy CELP decoders

The Conformance criteria for Full Accuracy CELP decoder are based on a maximum absolute difference signal value and a maximum RMS value over the full length of the difference signal.

To be called an ISO/IEC 14496-3 Full Accuracy CELP decoder, the decoder shall provide an output 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 $\frac{2^{-15}}{\sqrt{12}}$. In addition, the difference signal shall have a maximum absolute value of at most 2^{-14} relative to full scale.

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.

5.5.12.6 Descriptions of the audio test bitstreams

Table 5-22 — Test Bitstreams for the CELP object: MPE modes

File Name	CE00	CE01	CE02	CE03	CE04	CE05	CE06
Supplier	NEC	NEC	NEC	NEC	NEC	NEC	NEC
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 5-23 — Test Bitstreams for the CELP object: RPE modes

File Name	CE07	CE08	CE09	CE10	CE11	CE12	CE13	CE14
Supplier	Philips	Philips	Philips	Philips	Philips	Philips	Philips	Philips
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

5.5.13 HVXC

The HVXC object 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.

5.5.13.1 DecoderSpecificInfo Characteristics

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

d) AudioObjectType
e) samplingFrequencyIndex
f) channelConfiguration
g) HVXCrateMode
h) HVXCvarMode

5.5.13.2 Audio Access Unit Characteristics

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

5.5.13.3 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 5-24 — 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.

5.5.13.4 Procedure to Test Bitstream Conformance

5.5.13.4.1 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).

5.5.13.4.2 Audio Access Unit.

No restrictions to the Audio Access Unit.

5.5.13.5 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.

5.5.13.6 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 5-2 shows the decoder output signal timing for testing decoder conformamce.

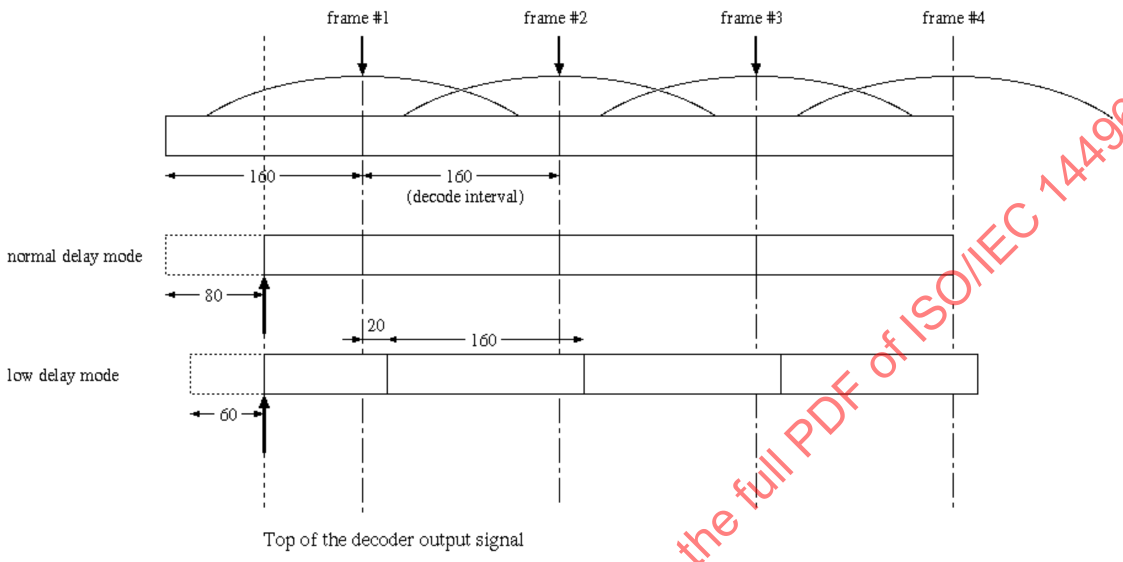

Figure 5-2 — Decoder output signal timing for testing decoder conformance

5.5.13.6.1 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.

5.5.13.6.1.1 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 Figures 4 and 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.

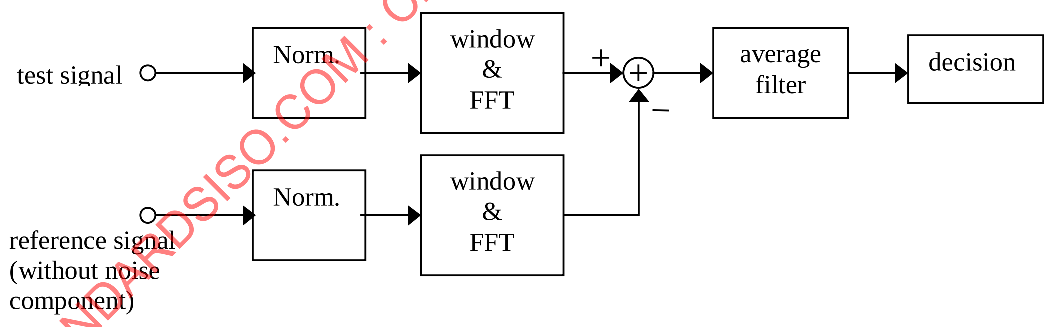

Figure 5-3 — Block Diagram of the Conformance Testing Procedure (All Voiced bitstream)

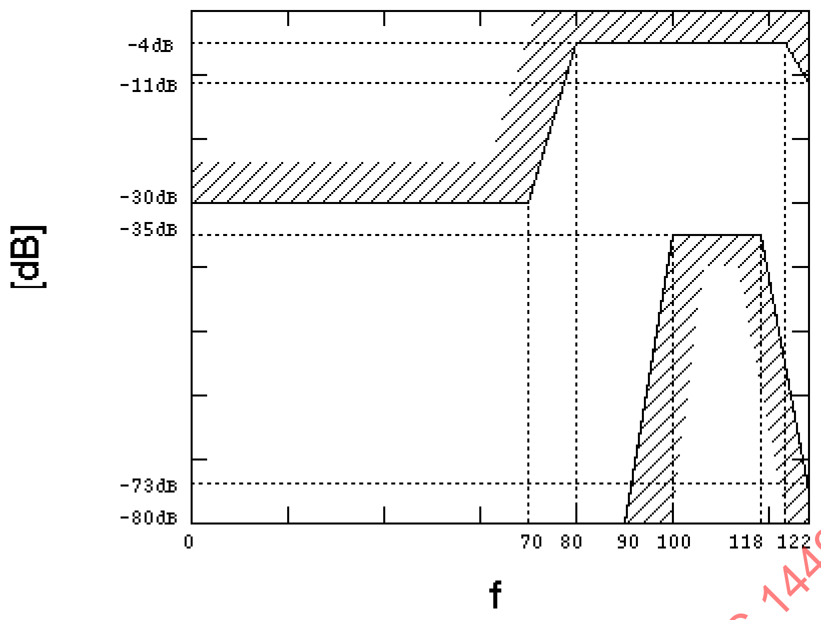

Figure 5-4 — Acceptable range of differential spectrum (for bitstream HV01)

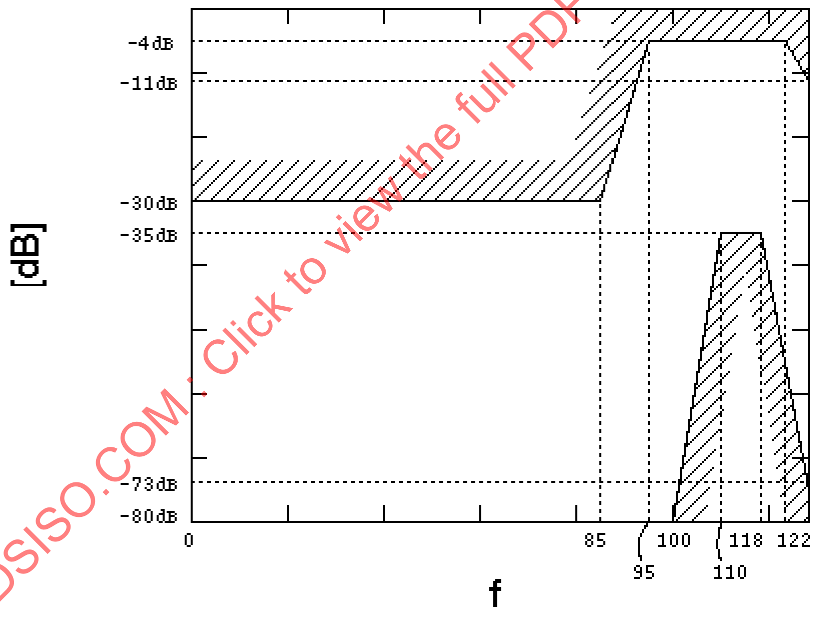

Figure 5-5 — Acceptable range of differential spectrum (for bitstream HV02)

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”);
      }
    }
}
```

5.5.13.6.1.2 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 5.5.1.6.2, 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

5.5.13.6.2 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 5.5.1.6.2) with random number generators disabled.
- Harmonic phase initialization (verification of phase randomness)

5.5.13.6.2.1 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 5.5.1.6.2) 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

5.5.13.6.2.2 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 5-6. For successive two “Voiced” frames, three segments are obtained.

2. For each segment, peak is searched, rms value and “peak/rms” value are computed.

3. 1. and 2. are repeated for a whole decoder output signal.

4. Average and deviation of “peak/rms” value are computed.

If the obtained average and deviation of “peak/rms” is within a range shown in Table 5-25, it can be said that the decoder under the test satisfies the conformance condition.

The above testing procedure is executed using dedicated software.

Table 5-25 — 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]

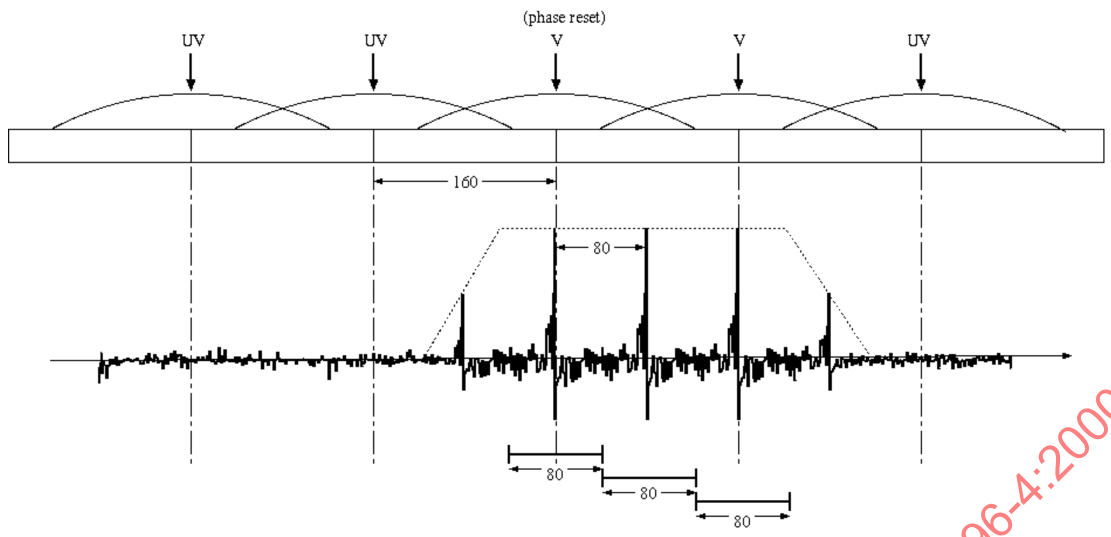

Figure 5-6 — An example decoder output signal and HVXC framing timing

5.5.13.7 Descriptions of the audio test bitstreams

Table 5-26 — HVXC object type test bitstream

File Name	HV01	HV02	HV03	HV04
Supplier	Sony	Sony	Sony	Sony
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 5-26 — HVXC object type test bitstream (continued)

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

5.5.13.8 Descriptions of the audio reference output signal

Table 5-27 — HVXC object type reference output signal

File Name	HV01ref1	HV01ref2	HV01ref3	HV01ref4	HV01ref5
Supplier	Sony	Sony	Sony	Sony	Sony
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 5-27 — HVXC object type reference output signal (continued)

File Name	HV01ref6	HV01ref7	HV01ref8	HV01ref9	HV01ref10
Supplier	Sony	Sony	Sony	Sony	Sony
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 5-27 — HVXC object type reference output signal (continued)

File Name	HV02ref1	HV02ref2	HV03ref1	HV03ref2	HV04ref1	HV04ref2
Supplier	Sony	Sony	Sony	Sony	Sony	Sony
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 5-27 — HVXC object type reference output signal (continued)

File Name	HV05ref1	HV05ref2	HV05ref3	HV05ref4	HV05ref5
Supplier	Sony	Sony	Sony	Sony	Sony
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 5-27 — HVXC object type reference output signal (continued)

File Name	HV05ref6	HV05ref7	HV05ref8	HV05ref9	HV05ref10
Supplier	Sony	Sony	Sony	Sony	Sony
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 5-27 — HVXC object type reference output signal (continued)

File Name	HV06ref1	HV06ref2	HV07ref1	HV07ref2
Supplier	Sony	Sony	Sony	Sony
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

5.5.14 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 5-7 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 5-7 are distinguished in the Table 5-28.

Table 5-28 — 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

5.5.14.1 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.

5.5.14.2 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

5.5.14.3 Procedure to Test Bitstream Conformance

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

5.5.14.4 Decoder Characteristics

Decoder conformance test will be applied to several interfaces identified in the Table 5-28. 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.

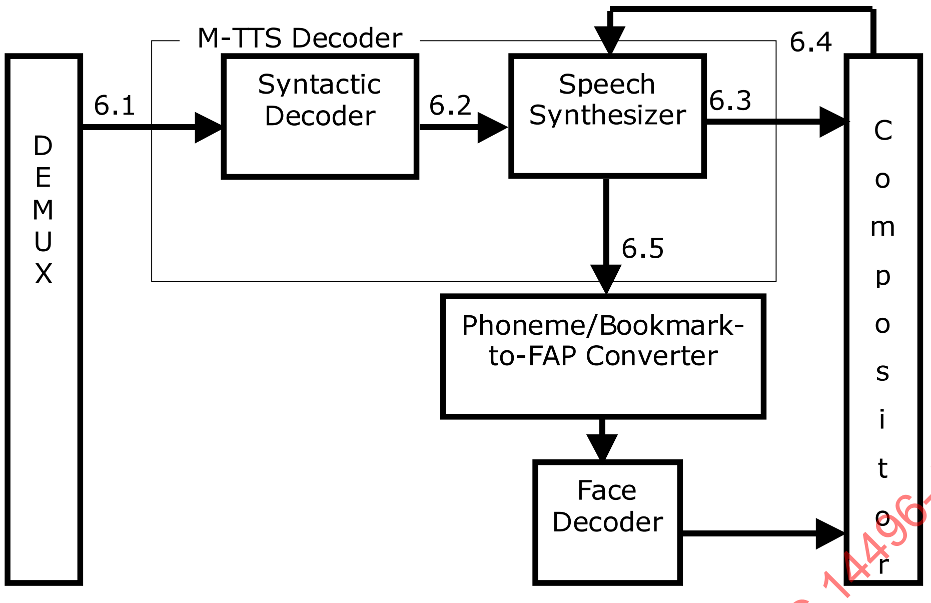

Figure 5-7 — MPEG-4 Audio TTS decoder architecture. (Text-to-Speech in ISO/IEC 14496-3)

5.5.14.5 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 29. 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 29. 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.

5.5.14.6 Descriptions of the test bitstreams

ISO/IEC 14496-3 (MPEG-4) audio test bitstreams are suggested as follows:

Table 5-29 — TTS Object Test Bitstream

File Name	TTS1	TTS2	TTS3	TTS4	TTS5	TTS6	TTS7	TTS8
Supplier	ETRI	ETRI	ETRI	ETRI	ETRI	ETRI	ATT	ATT
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.

5.5.15 General MIDI

The General MIDI object 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.

5.5.15.1 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.

5.5.15.1.1 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.

5.5.15.1.2 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.

5.5.15.2 Conformance

The General MIDI object is included only to provide interoperability with existing content. Normative sound quality and decoder behaviour are not provided with the General MIDI object. 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.

5.5.16 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 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)

5.5.16.1 Procedure to Test Decoder Conformance

5.5.16.1.1 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.

5.5.16.1.2 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.

5.5.16.2 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.

5.5.17 Algorithmic Synthesis and AudioFX

The Algorithmic Synthesis object 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.

5.5.17.1 DecoderSpecificInfo Characteristics

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

a) **orchestra** block

5.5.17.2 Audio Access Unit Characteristics

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

5.5.17.3 Procedure to Test Bitstream Conformance

5.5.17.3.1 DecoderSpecificInfo Characteristics

AudioObjectType: Shall be encoded with the value 16

SamplingFrequencyIndex: Shall be encoded with the following values:

Table 5-30

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 5-31

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 5-32

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.

5.5.17.3.2 Audio Access Unit Characteristics

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

5.5.17.4 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 5-33, called « Algorithmic Synthesis Complexity Values for Levels»,specifies values for SA Object type 3 for algorithmic synthesis: they are used in Synthesis Audio Profile Level definitions to define "Low", "Medium" and "High" Complexity values:

Table 5-33 — 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 5.6.7 and Table 5-34, where these values are used.

Table 5-34 — 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

5.5.17.5 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 “RMS Measurement.” 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.To be called an ISO/IEC 14496-3 audio decoder, the decoder shall provide an output 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^{-15}/sqrt(12). In addition, the difference signal shall have a maximum absolute value of at most 2^{-14} relative to full-scale. 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 5-33. See also subclause 5.5.17.4 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 5-33. 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 5.5.17.4. 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 5.5.17.6. 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 the following figure:

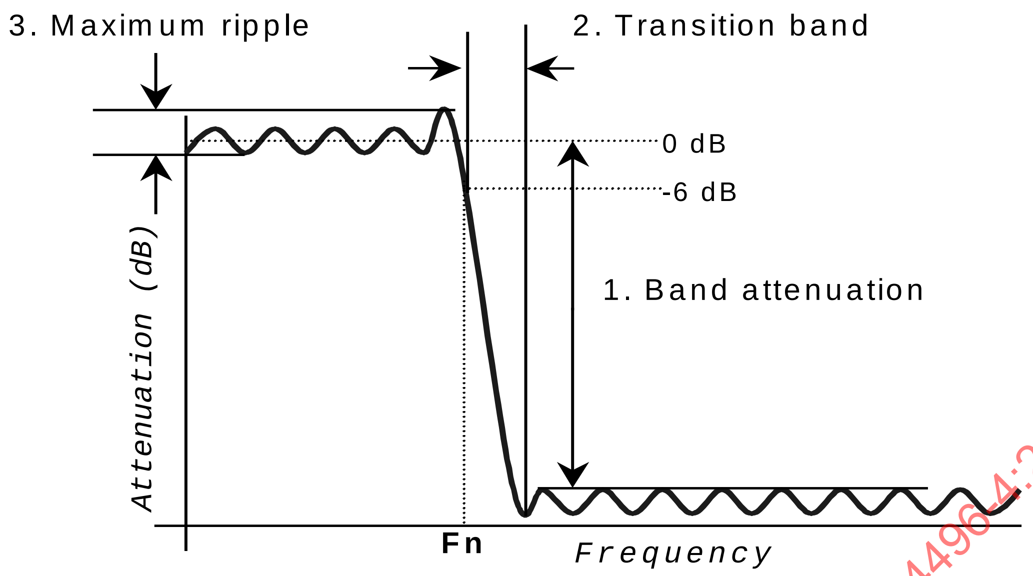

Figure 5-8

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. Transition band 15% of Fn; 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."

5.5.17.6 Descriptions of Conformance Bitstreams

All conformance bitstreams whose memory requirements and processing level, as indicated in Table 5-35, 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"

Exrecises 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 5.5.17.5 for test.

Bitstream SY013 "filters.mp4"

Exrecises 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 5-35 — Algorithmic Synthesis and Audio Fx Object Type Test Bitstreams

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

Table 5-35 — Algorithmic Synthesis and Audio Fx Object Type Test Bitstreams (continued)

File Name	SY008	SY009	SY010	SY011	SY012	SY013
Supplier	EPFL	MIT/ EPFL	EPFL	EPFL	EPFL	EPFL
Content	mood	PC	sin1	sin2	noise	filters
RCU - RAM (KB)	Med	*> High*	-	-	-	
Processing level	3520	40	-	-	-	

5.5.18 Main Synthetic

The main synthetic object 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.

5.5.18.1 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.

5.5.18.2 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.

5.5.18.3 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.

5.5.18.3.1 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.

5.5.18.3.2 Audio Access Unit Characteristics

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

5.5.18.4 Procedure to Test Decoder Conformance

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

- MIDI
- Wavetable synthesis
- Algorithmic synthesis and AudioFX

5.5.18.5 Descriptions of Conformance Bitstreams

See sections on the following audio objects:

- MIDI
- Wavetable synthesis
- Algorithmic synthesis and AudioFX

5.6 Audio Composition

5.6.1 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).

5.6.1.1 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 5-36 — 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 Objects: number of possible spatialized channels
- AudioFX: see subclause 5.5.17
- Reaction Time of a BIFS field update: the maximum time in msec. until the changes is audible.

5.6.1.2 Levels for Systems Audio Scene Graph Profile

Following these considerations, audio composition Levels are defined in the form of the following table:

Table 5-37 — 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 5-34)	Low Complexity (Table 5-34)	Medium Complexity (Table 5-34)	High Complexity (Table 5-34)
Spatialization	0	4	16	32

5.6.1.3 Composition Unit Inputs

Table 5-38

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.

5.6.1.4 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 5.6.3 to 5.6.7. Test scenes are defined using composition units described in subclause 5.6.1.3 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 5.6.1.3, 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.

5.6.2 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.

5.6.2.1 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.

5.6.2.2 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.

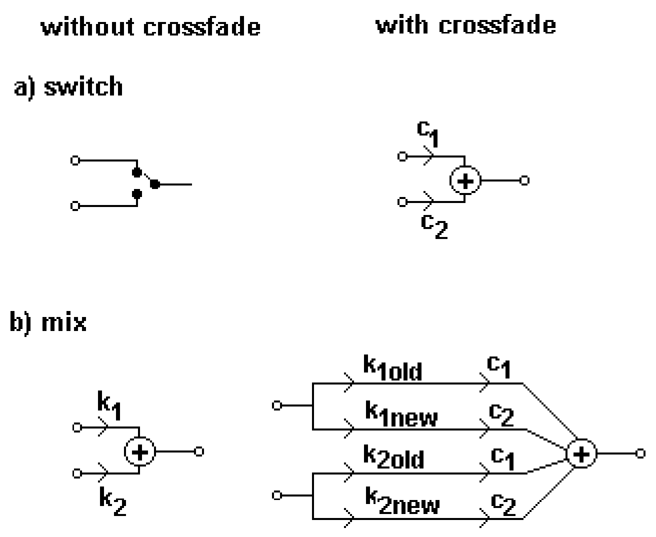

Figure 5-9 — Click free switch and mix

The above explained cross fade applies to the nodes **AudioSource**, **AudioMix**, **AudioSwitch**, and **AudioClip**.

5.6.2.3 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.

5.6.3 AudioSource and Sound2D

5.6.3.1 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).

5.6.3.2 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 of the residual after comparison to the reference signal. 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 RMS level of the difference signal between the output of the decoder under test and the supplied reference output is less than 2^{-15}/sqrt(12). In addition, the difference signal shall have a maximum absolute value of at most 2^{-14} relative to full-scale. 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.

5.6.3.3 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 5-10 — 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 5-10, 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 5-39 — AudioBIFS Test Bitstream

File Name	AB001	AB002	AB003	AB004	AB005	AB006
Supplier	EPFL	EPFL	EPFL	EPFL	EPFL	EPFL
Content	BIFS	BIFS	BIFS	BIFS	BIFS	BIFS
Bitstream from a source (url)	CU1	CU2	CU3	CU4	CU1	CU4

5.6.4 AudioSource and Sound

5.6.4.1 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).

5.6.4.2 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 described in the subclause “RMS Measurement.” Software is provided for performing this verification procedure. To be called an ISO/IEC 14496-3 audio decoder, the decoder shall provide an output 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^{-15}/sqrt(12). In addition, the difference signal shall have a maximum absolute value of at most 2^{-14} relative to full-scale. 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.

5.6.4.3 Audio BIFS Test Scenes

AB011 One AudioSource node connected to one Sound node with default fields, except spatialize = FALSE, using CU1_Yx as input.

AB012, AB013, AB014 The same as AB011, with CU2_Yx, CU3_Yx, CU4_Yx as inputs, respectively.

AB015 One AudioSource node connected to one Sound node with default fields, except location, using CU1_Yx as input. The sound position describes an arch at a distance of 2 meters from the listener, moving from -60° to 60° in azimuth. Heigth is constant at 2 meters for both, the Sound location and ListeningPoint. The location field is updated every 0.5 seconds and the source is moved by 15° clockwise at each update.

AB016 The same as AB005 using CU4_Yx as input.

For sequences AB011 to AB016 the electronic attachment to this part of ISO/IEC 14496 provides both a normative MP4 file and a textual parametric source, to be encoded by the decoder provider using the specific input CU and either the reference encoder or an equivalent. For sequences AB011 to AB014 the electronic attachment to this part of ISO/IEC 14496 provides reference output.

Table 5-40 — AudioBIFS Test Bitstream

File Name	AB011	AB012	AB013	AB014	AB015	AB016
Supplier	EPFL	EPFL	EPFL	EPFL	EPFL	EPFL
Content	BIFS	BIFS	BIFS	BIFS	BIFS	BIFS
Bitstream from a source (url)	CU1	CU2	CU3	CU4	CU1	CU4

5.6.5 AudioSwitch

See also subclause 5.6.2.2. Conformance Test of the AudioSwitch node is not required for decoders at Level 1.

5.6.5.1 BIFS fields Characteristic

None.

5.6.5.2 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.

5.6.5.3 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 5-41

File Name	AB031
Supplier	EPFL
Content	BIFS
Bitstream 1 from a source (url)	CU1
Bitstream 2 from a source (url)	CU1

5.6.6 AudioMix and Sampling Rate Conversion

See also subclause 5.6.2.2.

5.6.6.1 BIFS fields Characteristic

None.

5.6.6.2 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 5.6.2.3) Software is provided for performing this verification procedure. To be called an ISO/IEC 14496-3 audio 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.

5.6.6.3 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 5-42

File Name	AB041	AB042	AB043	AB044
Supplier	EPFL	EPFL	EPFL	EPFL
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

5.6.7 AudioFX

See also subclause 5.5.17

5.6.7.1 BIFS fields Characteristic

Restrictions on field values.

5.6.7.2 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 described in the subclause "RMS Measurement." Software is provided for performing this verification procedure. To be called an ISO/IEC 14496-3 audio decoder, the decoder shall provide an output 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^{-15}/sqrt(12). In addition, the difference signal shall have a maximum absolute value of at most 2^{-14} relative to full-scale. This test only verifies the computational accuracy of an implementation.

5.6.7.3 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 5-43 — AudioBIFS Test Bitstream

File Name	AB101	AB102
Supplier	EPFL	EPFL
Content	Stripe	Stripe
Orchestra definition (orch)	Stripe	Stripe
Score definition (score)	Stripe	Stripe
Bitstream 1 from a source (url)	CU1	CU4

6 DMIF

6.1 Introduction

This clause defines compliance to ISO/IEC 14496-6 Delivery Multimedia Integration Framework (DMIF) 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 6.1 of this document in the form of Protocol Implementation Conformance Statement (PICS) proforma. The abstract test cases used for dynamic review are described in subclause 6.2.

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 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 subclause.

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 signaling 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.

6.2 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.

6.2.1 Global statement of conformance

Table 6-1 — 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
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					

6.2.2 DMIF Signalling

6.2.2.1 DMIF Signalling Conformance

DMIF V1 specifies the DMIF Default Signalling Protocol, its messages and their mapping into various native network signalling protocols, such as ATM Q.2931 in case of ATM.

An implementation may be compliant with any or multiple actual DMIF Signalling protocols.

6.2.2.2 Primitive support for DS

Table 6-2 — PICS for DMIF signaling PDU Support

Item	Does the implementation support ...	ISO/IEC 14496-6 reference	Condition for the status	Status	Implemented? Y=Yes, N=No, n/a=not applicable
Pdu1	DS_SessionSetupRequest	12.1.2.1	DS Protocol	c.m	
Pdu2	DS_SessionSetupConfirm	12.1.2.2	Pdu1	c.m	
Pdu3	DS_SessionReleaseRequest	12.1.2.3	Pdu1	c.m	
Pdu4	DS_SessionReleaseConfirm	12.1.2.4	Pdu3	c.m	
Pdu5	DS_ServiceAttachRequest	12.1.2.5	DS Protocol	c.m	
Pdu6	DS_ServiceAttachConfirm	12.1.2.6	Pdu5	c.m	
Pdu7	DS_ServiceDetachRequest	12.1.2.7	Pdu5	c.m	
Pdu8	DS_ServiceDetachConfirm	12.1.2.8	Pdu7	c.m	
Pdu9	DS_TransMuxSetupRequest	12.1.2.9	DS Protocol	c.m	
Pdu10	DS_TransMuxSetupConfirm	12.1.2.10	Pdu9	c.m	
Pdu11	DS_TransMuxReleaseRequest	12.1.2.11	Pdu9	c.m	
Pdu12	DS_TransMuxReleaseConfirm	12.1.2.12	Pdu11	c.m	
Pdu13	DS_ChannelAddRequest	12.1.2.13	DS Protocol	c.m	
Pdu14	DS_ChannelAddConfirm	12.1.2.14	Pdu13	c.m	
Pdu15	DS_ChannelAddedRequest	12.1.2.15	DS Protocol	c.m	
Pdu16	DS_ChannelAddedConfirm	12.1.2.16	Pdu15	c.m	
Pdu17	DS_ChannelDeleteRequest	12.1.2.17	Pdu15	c.m	
Pdu18	DS_ChannelDeleteConfirm	12.1.2.18	Pdu17	c.m	
Pdu19	DS_TransMuxConfigRequest	12.1.2.19	DS Protocol	c.m	
Pdu20	DS_TransMuxConfigConfirm	12.1.2.20	Pdu19	c.m	
Pdu21	DS_UserCommand	12.1.2.21	DS Protocol	c.m	

6.2.2.3 Parameter Support for DS Protocol

Table 6-3 — Parameter Support for DS Session Messages

Item	Parameter	ISO/IEC 14496-6 reference	Condition for the status	Status	Values allowed
DS_SessionSetupRequest					
Par1	dsmccMessageHeader()	12.1.2.1		m	see above
Par2	networkSessionId	12.1.2.1		m	same as SessionId in subclause 4.3 of ISO/IEC 13818-6
Par3	compatibilityDescriptor()	12.1.2.1		m	see above
DS_SessionSetupConfirm					
Par4	dsmccMessageHeader()	12.1.2.2		m	see above
Par5	response	12.1.2.2		m	see ISO/IEC 14496-6 subclause 11.2.7

Table 6-3 — Parameter Support for DS Session Messages (continued)

Item	Parameter	ISO/IEC 14496-6 reference	Condition for the status	Status	Values allowed
Par6	compatibilityDescriptor()	12.1.2.2		m	see above
DS_SessionReleaseRequest					
Par7	dsmccMessageHeader()	12.1.2.3		m	see above
Par8	networkSessionId	12.1.2.3		m	same as SessionId in subclause 4.3 of ISO/IEC 13818-6
Par9	reason	12.1.2.3		m	see ISO/IEC 14496-6 subclause 11.2.6
DS_SessionReleaseConfirm					
Par10	dsmccMessageHeader()	12.1.2.4		m	see above
Par11	response	12.1.2.4		m	see ISO/IEC 14496-6 subclause 11.2.7
DS_ServiceAttachRequest					
Par12	dsmccMessageHeader()	12.1.2.5		m	see above
Par13	networkSessionId	12.1.2.5		m	same as SessionId in subclause 4.3 of ISO/IEC 13818-6
Par14	serviceId	12.1.2.5		m	see ISO/IEC 14496-6 subclause 11.3
Par15	serviceNameLen	12.1.2.5		m	0x00 - 0xff
Par16	serviceName	12.1.2.5		m	see ISO/IEC 14496-6 subclause 10.3
Par17	ddData()	12.1.2.5		m	see ISO/IEC 14496-6 subclause 11.2.2
DS_ServiceAttachConfirm					
Par18	dsmccMessageHeader()	12.1.2.6		m	see above
Par19	response	12.1.2.6		m	see ISO/IEC 14496-6 subclause 11.2.7
Par20	ddData()	12.1.2.6		m	see ISO/IEC 14496-6 subclause 11.2.2
DS_ServiceDetachRequest					
Par21	dsmccMessageHeader()	12.1.2.7		m	see above
Par22	networkSessionId	12.1.2.7		m	same as SessionId in subclause 4.3 of ISO/IEC 13818-6
Par23	serviceId	12.1.2.7		m	see ISO/IEC 14496-6 subclause 11.3
Par24	reason	12.1.2.7		m	see ISO/IEC 14496-6 subclause 11.2.6
DS_ServiceDetachConfirm					
Par25	dsmccMessageHeader()	12.1.2.8		m	see above
Par26	response	12.1.1.8		m	see ISO/IEC 14496-6 subclause 11.2.7

Table 6-3 — Parameter Support for DS Session Messages (continued)

Item	Parameter	ISO/IEC 14496-6 reference	Condition for the status	Status	Values allowed
DS_TransMuxSetupRequest					
Par27	dsmccMessageHeader()	12.1.2.9		m	see above
Par28	networkSessionId	12.1.2.9		m	same as SessionId in subclause 4.3 of ISO/IEC 13818-6
Par29	count	12.1.2.9		m	0x00 - 0xff
Par30	TAT	12.1.2.9		m	see ISO/IEC 14496-6 subclause 11.3
Par31	qosDescriptor()	12.1.2.9		m	see ISO/IEC 14496-6 subclause 11.2.4
Par32	resources()	12.1.2.9		m	see ISO/IEC 14496-6 subclause 11.2.3
DS_TransMuxSetupConfirm					
Par33	dsmccMessageHeader()	12.1.2.10		m	see above
Par34	count	12.1.2.10		m	0x00 - 0xff
Par35	response	12.1.2.10		m	see ISO/IEC 14496-6 subclause 11.2.7
Par36	resources()	12.1.2.10		m	see ISO/IEC 14496-6 subclause 11.2.3
DS_TransMuxReleaseRequest					
Par37	dsmccMessageHeader()	12.1.2.11		m	see above
Par38	networkSessionId	12.1.2.11		m	same as SessionId in subclause 4.3 of ISO/IEC 13818-6
Par39	count	12.1.2.11		m	0x00 - 0xff
Par40	TAT	12.1.2.11		m	see ISO/IEC 14496-6 subclause 11.3
DS_TransMuxReleaseConfirm					
Par41	dsmccMessageHeader()	12.1.2.12		m	see above
Par42	count	12.1.2.12		m	0x00 - 0xff
Par43	response	12.1.2.12		m	see ISO/IEC 14496-6 subclause 11.2.7
DS_ChannelAddRequest					
Par44	dsmccMessageHeader()	12.1.2.13		m	see above
Par45	networkSessionId	12.1.2.13		m	same as SessionId in subclause 4.3 of ISO/IEC 13818-6
Par46	serviceId	12.1.2.13		m	see ISO/IEC 14496-6 subclause 11.3
Par47	count	12.1.2.13		m	0x00 - 0xff
Par48	CAT	12.1.2.13		m	see ISO/IEC 14496-6 subclause 11.3
Par49	direction	12.1.2.13		m	see ISO/IEC 14496-6 subclause 11.2.5
Par50	qosDescriptor()	12.1.2.13		m	see ISO/IEC 14496-6 subclause 11.2.4

Table 6-3 — Parameter Support for DS Session Messages (continued)

Item	Parameter	ISO/IEC 14496-6 reference	Condition for the status	Status	Values allowed
Par51	ddData()	12.1.2.13		m	see ISO/IEC 14496-6 subclause 11.2.2
DS_ChannelAddConfirm					
Par52	dsmccMessageHeader()	12.1.2.14		m	see above
Par53	count	12.1.2.14		m	0x00 - 0xff
Par54	response	12.1.2.14		m	see ISO/IEC 14496-6 subclause 11.2.7
Par55	TAT	12.1.2.14		m	see ISO/IEC 14496-6 subclause 11.3
Par56	ddData()	12.1.2.14		m	see ISO/IEC 14496-6 subclause 11.2.2
DS_ChannelAddedRequest					
Par57	dsmccMessageHeader()	12.1.2.15		m	see above
Par58	networkSessionId	12.1.2.15		m	same as SessionId in subclause 4.3 of ISO/IEC 13818-6
Par59	serviceId	12.1.2.15		m	see ISO/IEC 14496-6 subclause 11.3
Par60	count	12.1.2.15		m	0x00 - 0xff
Par61	CAT	12.1.2.15		m	see ISO/IEC 14496-6 subclause 11.3
Par62	direction	12.1.2.15		m	see ISO/IEC 14496-6 subclause 11.2.5
Par63	qosDescriptor()	12.1.2.15		m	see ISO/IEC 14496-6 subclause 11.2.4
Par64	TAT	12.1.2.15		m	see ISO/IEC 14496-6 subclause 11.3
Par65	ddData()	12.1.2.15		m	see ISO/IEC 14496-6 subclause 11.2.2
DS_ChannelAddedConfirm					
Par66	dsmccMessageHeader()	12.1.2.16		m	see above
Par67	count	12.1.2.16		m	0x00 - 0xff
Par68	response	12.1.2.16		m	see ISO/IEC 14496-6 subclause 11.2.7
Par69	ddData()	12.1.2.16		m	see ISO/IEC 14496-6 subclause 11.2.2
DS_ChannelDeleteRequest					
Par70	dsmccMessageHeader()	12.1.2.17		m	see above
Par71	networkSessionId	12.1.2.17		m	same as SessionId in subclause 4.3 of ISO/IEC 13818-6
Par72	count	12.1.2.17		m	0x00 - 0xff
Par73	CAT	12.1.2.17		m	see ISO/IEC 14496-6 subclause 11.3
Par74	reason	12.1.1.17		m	see ISO/IEC 14496-6 subclause 11.2.6
DS_ChannelDeleteConfirm					
Par75	dsmccMessageHeader()	12.1.2.18		m	see above
Par76	count	12.1.2.18		m	0x00 - 0xff

Table 6-3 — Parameter Support for DS Session Messages (continued)

Item	Parameter	ISO/IEC 14496-6 reference	Condition for the status	Status	Values allowed
Par77	response	12.1.2.18		m	see ISO/IEC 14496-6 subclause 11.2.7
DS_TransMuxConfigRequest					
Par78	dsmccMessageHeader()	12.1.2.19		m	see above
Par79	networkSessionId	12.1.2.19		m	same as SessionId in subclause 4.3 of ISO/IEC 13818-6
Par80	count	12.1.2.19		m	0x00 - 0xff
Par81	TAT	12.1.2.19		m	see ISO/IEC 14496-6 subclause 11.3
Par82	ddData()	12.1.2.19		m	see above
DS_TransMuxConfigConfirm					
Par83	dsmccMessageHeader()	12.1.2.20		m	see above
Par84	count	12.1.2.20		m	0x00 - 0xff
Par85	response	12.1.2.20		m	see ISO/IEC 14496-6 subclause 11.2.7
DS_UserCommand					
Par86	dsmccMessageHeader()	12.1.2.21		m	see above
Par87	networkSessionId	12.1.2.21		m	same as SessionId in subclause 4.3 of ISO/IEC 13818-6
Par88	count	12.1.2.21		m	0x00 - 0xff
Par89	CAT	12.1.2.21		m	see ISO/IEC 14496-6 subclause 11.3
Par90	ddData()	12.1.2.21		m	see above

6.2.2.4 Parameter support for DSM-CC User-to-Network Message Header

Table 6-4 — DSM-CC User-to-Network Message Header Parameters

Item	Parameter	ISO/IEC 13818-6 reference	Status	Values allowed
Par1	ProtocolDiscriminator	2	m	0x11
Par2	DsmccType	2	m	0x07
Par3	MessageId	ISO/IEC 14496-6	m	
Par4	TransactionId	2	m	0x00 - 0x03
Par5	AdaptationLength	2	m	0x00 - 0xFF
Par6	MessageLength	2	m	0x0000-0xFFFF
Par7	DsmccAdaptationHeader	2.1	o	
Par7.1	AdaptationType	2.1	c:m c:o	0x01, 0x02 0x80-0xFF
Par7.1.2	Reserved	2.1.1	c:m	0xFF
Par7.1.3	CaSystemId	2.1.1	c:m	(CA_system_ID in ISO/IEC 13818-1)
Par7.1.4	ConditionalAccessLength	2.1.1	c:m	TBA
Par7.1.5	ConditionalAccesDataByte	2.1.1	c:m	TBA
Par7.2	DsmccUserId	2.1.2	o	
Par7.2.1	Reserved	2.1.2	c:m	0xFF
Par7.2.2	UserId	2.1.2	c:m	(values of clientId or serverId in U-N session messages)

6.2.2.5 Parameter support for DSM-CC Compatibility Descriptors

Table 6-5 — DSM-CC Compatibility Descriptor Parameters

Item	Parameter	ISO/IEC 13818-6 reference	Status	Values allowed
Par1	CompatibiltyDescriptorLength	6.1	m	0x0000-0xFFFF
Par2	DescriptorCount	6.1	m	0x0000-0xFFFF
Par3	DescriptorType	6.1	m o	0x00 - 0x02 0x40 - 0xFF
Par4	DescriptorLength	6.1	m	0x00 - 0xFF
Par5	SpecifierType	6.1	m o	0x01 0x80 - 0xFF
Par6	SpecifierData	6.1	m	0x000000 - 0xFFFFFF
Par7	Model	6.1	m	0x0000- 0xFFFF
Par8	Version	6.1	m	0x0000- 0xFFFF
Par9	SubDescriptorCount	6.1	m	0x00 - 0xFF
Par10	SubDescriptor	6.1	m	0x00 - 0xFF
Par10.1	SubDescriptorType	6.1	m	0x00 - 0xFF
Par10.2	SubDescriptorLength	6.1	m	0x00 - 0xFF
Par10.3	AdditionalInformation	6.1	m	0x00 - 0xFF

6.2.3 Q.2931 Extensions for DMIF

This subclause holds when DMIF is operated over B-ISDN

6.2.3.1 Primitive Support for Q.2931 Extension for DMIF

Commands are per ITU-T Recommendation Q.2931: 1995, B-ISDN User Network Interface Layer 3 Specificatoin for Basic Call/Bearere Control.

6.2.3.2 Parameter Support for Q.2931 Extension for DMIF

The existing Q.2931 Parameters are supported in addition the extensions in Table 6-6 are used.

Table 6-6 — Parameter Support for Q.2931 Extensions

Item	Parameter	ITU-T Q.2931 Reference	Condition for the status	Status	Values allowed
First Q.2931 SETUP					
Par1	BHLI Type	TBA		m	0x00
Par2	DMIF_SelectorByte	12.1.2.1		m	ISO/IEC 14496-6, subclause 12.4.5.1
Par3	neworkSessionId	12.1.2.1		m	ISO/IEC 13818-6, subclause 4.3
Par4	Octet7-8	12.1.2.1		m	0x0000
Subsequent Q.2931 SETUP					
Par1	BHLI Type	TBA		m	0x00
Par2	DMIF_SelectorByte	12.1.2.1		m	ISO/IEC 14496-6, subclause 12.4.5.1
Par3	neworkSessionId	12.1.2.1		m	ISO/IEC 13818-6, subclause 4.3
Par4	TAT	12.1.2.1		m	0x0000 – 0xffff

6.3 The Conformance ATS

6.3.1 General

6.3.1.1 Introduction

This subclause contains the conformance abstract test suites (ATSs) 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.

6.3.1.2 Test Method

For conformance testing of the DMIF Signalling protocol, the Remote Test Method as defined in ISO/IEC 9646-1 and ISO/IEC 9646-2 is used. The characteristics of the Remote Test Method, shown in Figure 6-1, 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 6-1.

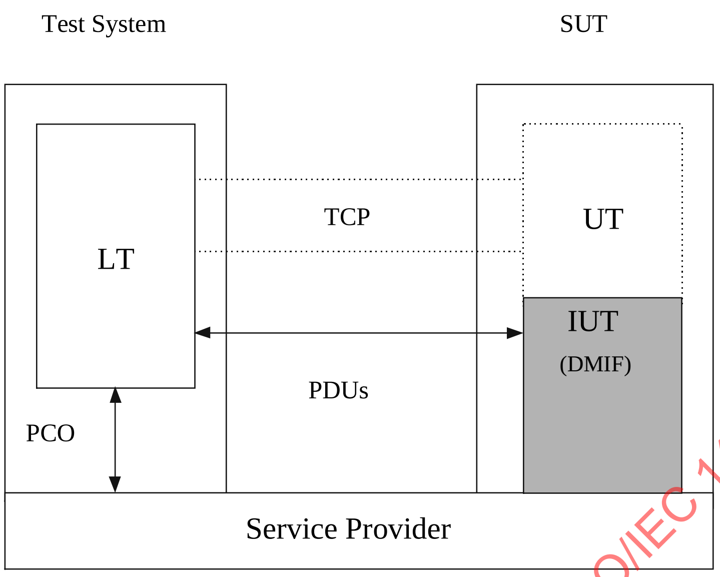

Figure 6-1 — The Remote Test Method

6.3.2 ATS for DS

6.3.2.1 Generic ATS for DS

6.3.2.1.1 Generic Test Environment

Figure 6-2 shows the function and role of the LT and the SUT in the test environment.

The UT is required only in some test case. The DMIF-Application Interface (DAI), whose semantic is defined in part 6 of this ISO/IEC 14496, is used as the interface between UT and IUT.

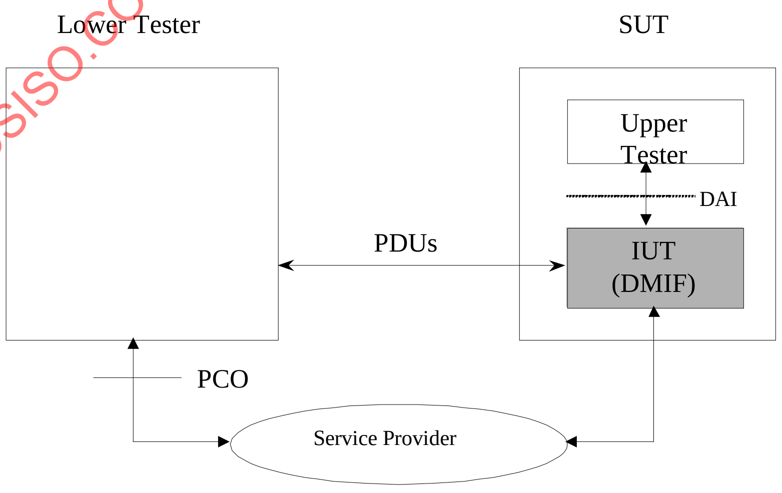

Figure 6-2 — The test method for the DMIF Signalling test environment

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.

6.3.2.1.2 Test Cases

6.3.2.1.2.1 DMIF Session primitives

Table 6-7 — DMIF Session Primitives Test Cases

Test Case #	Test Case Names	Reference to ISO/IEC 14496-6
1	Setting up a Session (originating side)	12.1.2.1, 12.1.2.2
2	Setting up a Session (destination side)	12.1.2.1, 12.1.2.2
3	Releasing a Session (originating side)	12.1.2.3, 12.1.2.4
4	Releasing a Session (destination side)	12.1.2.3, 12.1.2.4

6.3.2.1.2.1.1 Test Case 1 - Setting up a Session (originating side)

Test Purpose:

Verify that a Session Setup procedure is correctly performed at the originating side. This includes also a Service Attach procedure.

Test Preamble:

q) The UT requests through a DA_ServiceAttach(IN) to create a new service

r) The service is located at the LT

s) The SUT knows how to contact the LT

Test Procedure:

17 UT passes a DA_ServiceAttach(IN) to the IUT

18 IUT sends a 'DS_SessionSetupRequest' message

19 LT responds with a 'DS_SessionSetupConfirm' message

20 IUT sends a 'DS_ServiceAttachRequest' message

21 LT responds with a 'DS_ServiceAttachConfirm' message

22 IUT passes a DA_ServiceAttach(OUT) back to the UT

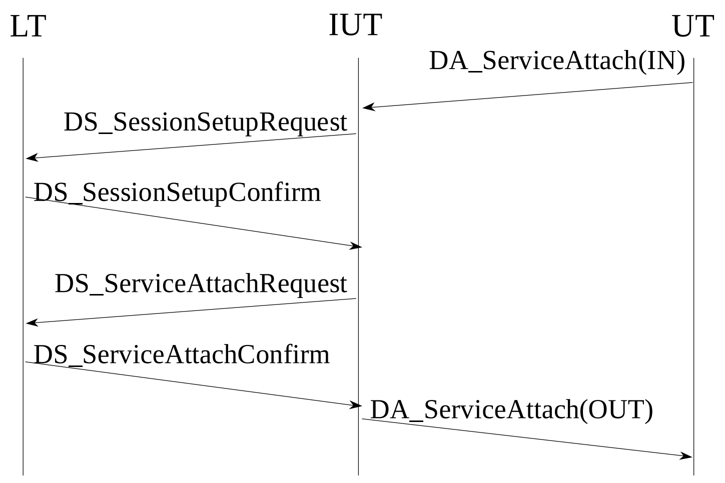

Figure 6-3 — Session Setup Origination side

Test Verdict:

Pass the test if the response to UT is:

Table 6-8 — Session Setup Origination Test Cases

Conditions	**Observations**
DA_ServiceAttach(IN) was valid DS_SessionSetupConfirm had response OK DS_ServiceAttachConfirm had response OK	DS_SessionSetupRequest has been originated consistently DS_ServiceAttachRequest has been originated consistently DA_ServiceAttach(OUT) has been generated consistently with response OK
DA_ServiceAttach(IN) was valid DS_SessionSetupConfirm had response OK DS_ServiceAttachConfirm had response not OK	DS_SessionSetupRequest has been originated consistently DS_ServiceAttachRequest has been originated consistently DA_ServiceAttach(OUT) has been generated consistently with response not OK and a Session Release procedure has been initiated
DA_ServiceAttach(IN) was valid DS_SessionSetupConfirm had response not OK	DS_SessionSetupRequest has been originated consistently DS_ServiceAttachRequest has not been originated DA_ServiceAttach(OUT) has been generated consistently with response not OK
DA_ServiceAttach(IN) was not valid	DS_SessionSetupRequest has not been originated DA_ServiceAttach(OUT) has been generated consistently with response not OK

6.3.2.1.2.1.2 Test Case 2 - Setting up a Session (destination side)

Test Purpose:

Verify that a Session Setup procedure is correctly performed at the destination side.

Test Preamble:

t) The LT knows how to contact the SUT

Test Procedure:

23 LT sends a 'DS_SessionSetupRequest' message.

24 IUT responds with a 'DS_SessionSetupConfirm' message

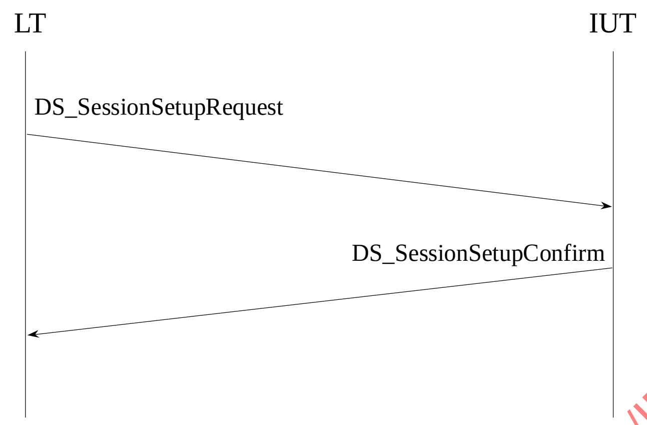

Figure 6-4 — Session Setup Destination side

Test Verdict:

Pass the test if the response to LT is:

Table 6-9 — Session Setup Destination Test Cases

Conditions	**Observations**
DS_SessionSetupRequest was valid	DS_SessionSetupConfirm has response OK
DS_SessionSetupRequest was not valid	DS_SessionSetupConfirm has response NOT OK

6.3.2.1.2.1.3 Test Case 3 - Releasing a Session (originating side)

Test Purpose:

Verify that a Session Release procedure is correctly performed at the originating side. This includes also a Service Detach procedure.

Test Preamble:

u) The UT requests through a DA_ServiceDetach(IN) to detach a service

v) The service was existing and operating at the LT, over an existing session

Test Procedure:

25 UT passes a DA_ServiceDetach(IN) to the IUT

26 IUT sends a 'DS_ServiceDetachRequest' message

27 LT responds with a 'DS_ServiceDetachConfirm' message

28 IUT sends a 'DS_SessionReleaseRequest' message

29 LT responds with a ‘DS_SessionReleaseConfirm’ message

30 IUT passes a DA_ServiceDetach(OUT) back to the UT

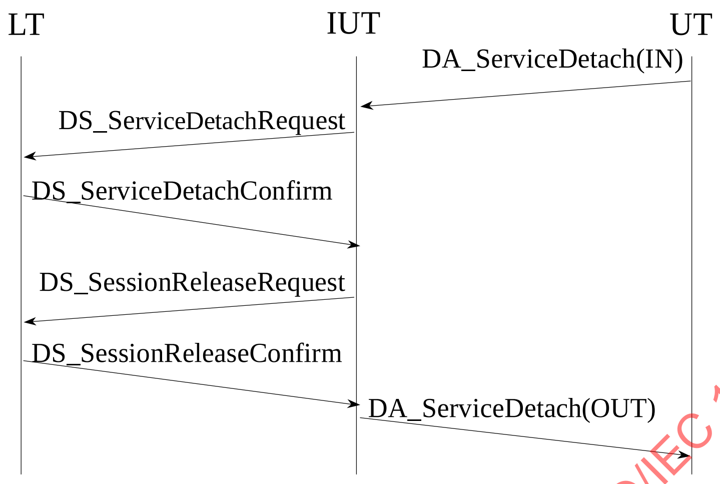

Figure 6-5 — Session Release Origination side

Test Verdict:

Pass the test if the response to UT is:

Table 6-10 — Session Release Origination Test Cases

Conditions	Observations
DA_ServiceDetach(IN) was valid DS_ServiceDetachConfirm had response OK DS_SessionReleaseConfirm had response OK	DS_ServiceDetachRequest has been originated consistently DS_SessionReleaseRequest has been originated consistently DA_ServiceDetach(OUT) has been generated consistently with response OK
DA_ServiceDetach(IN) was valid DS_ServiceDetachConfirm had response OK DS_SessionReleaseConfirm had response not OK	DS_ServiceDetachRequest has been originated consistently DS_SessionReleaseRequest has been originated consistently DA_ServiceDetach(OUT) has been generated consistently with response OK
DA_ServiceDetach(IN) was valid DS_ServiceDetachConfirm had response not OK	DS_ServiceDetachRequest has been originated consistently DS_SessionReleaseRequest has not been originated DA_ServiceDetach(OUT) has been generated consistently with response not OK
DA_ServiceDetach(IN) was not valid	DS_ServiceDetachRequest has not been originated DA_ServiceDetach(OUT) has been generated consistently with response not OK

6.3.2.1.2.1.4 Test Case 4 - Releasing a Session (destination side)

Test Purpose:

Verify that a Session Release procedure is correctly performed at the destination side.

Test Preamble:

w) The session was existing and operating between SUT and LT

Test Procedure:

31 LT sends a 'DS_SessionReleaseRequest' message.

32 IUT responds with a 'DS_SessionReleaseConfirm' message

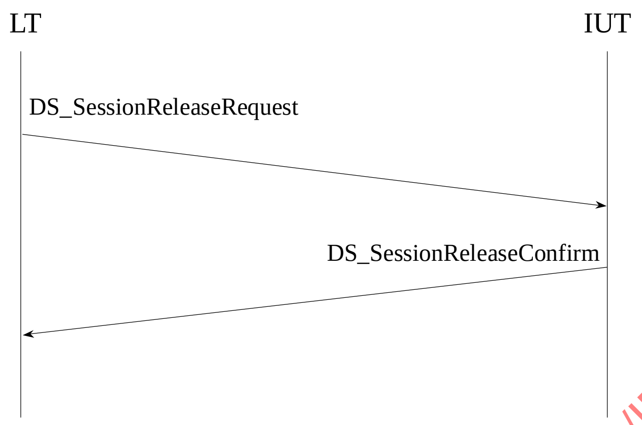

Figure 6-6 — Session Release Destination side

Test Verdict:

Pass the test if the response to LT is:

Table 6-11 — Session Release Destination Test Cases

Conditions	**Observations**
DS_SessionReleaseRequest was valid	DS_SessionReleaseConfirm has response OK
DS_SessionReleaseRequest was not valid	DS_SessionReleaseConfirm has response NOT OK

6.3.2.1.2.2 DMIF Service Primitives

Table 6-12 — DMIF DAI Service Primitives Test Cases

Test Case #	Test Case Names	Reference to ISO/IEC 14496-6
1	Attaching a Service (originating side)	12.1.2.5, 12.1.2.6
2	Attaching a Service (destination side)	12.1.2.5, 12.1.2.6
3	Detaching a Service (originating side)	12.1.2.7, 12.1.2.8
4	Detaching a Service (destination side)	12.1.2.7, 12.1.2.8

6.3.2.1.2.2.1 Test Case 1 - Attaching a Service (originating side)

Test Purpose:

Verify that a Service Attach procedure is correctly performed at the originating side.

Test Preamble:

x) The UT requests through a DA_ServiceAttach(IN) to create a new service

y) The service is located at the LT

z) A Session with the LT is already existing and operating

Test Procedure:

33 UT passes a DA_ServiceAttach(IN) to the IUT

34 IUT sends a 'DS_ServiceAttachRequest' message

35 LT responds with a 'DS_ServiceAttachConfirm' message

36 IUT passes a DA_ServiceAttach(OUT) back to the UT

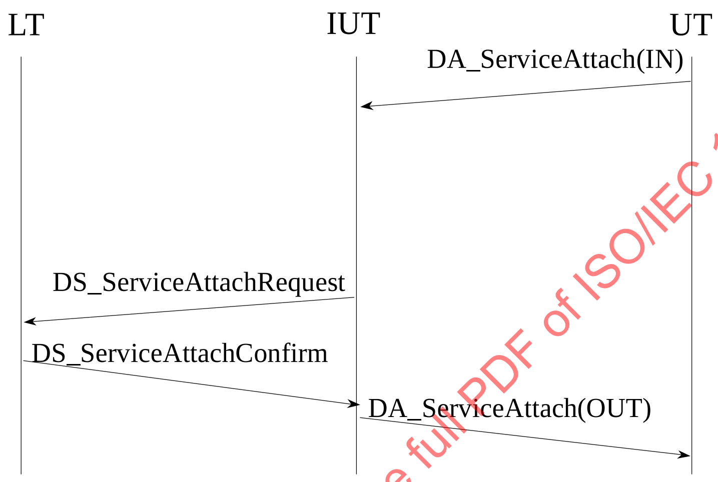

Figure 6-7 — Service Attach Origination side

Test Verdict:

Pass the test if the response to UT is:

Table 6-13 — Service Attach Origination Test Cases

Conditions	**Observations**
DA_ServiceAttach(IN) was valid DS_ServiceAttachConfirm had response OK	DS_ServiceAttachRequest has been originated consistently DA_ServiceAttach(OUT) has been generated consistently with response OK
DA_ServiceAttach(IN) was valid DS_ServiceAttachConfirm had response not OK	DS_ServiceAttachRequest has been originated consistently DA_ServiceAttach(OUT) has been generated consistently with response not OK
DA_ServiceAttach(IN) was not valid	DS_ServiceAttachRequest has not been originated DA_ServiceAttach(OUT) has been generated consistently with response not OK

6.3.2.1.2.2.2 Test Case 2 - Attaching a Service (destination side)

Test Purpose:

Verify that a Service Attach procedure is correctly performed at the destination side.

Test Preamble:

aa) A session is existing and operating between SUT and LT

Test Procedure:

37 LT sends a 'DS_ServiceAttachRequest' message.

38 IUT responds with a 'DS_ServiceAttachConfirm' message

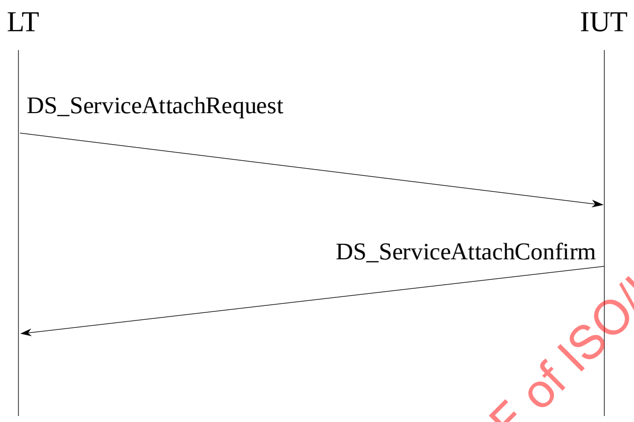

Figure 6-8 — Service Attach Destination side

Test Verdict:

Pass the test if the response to LT is:

Table 6-14 — Service Attach Destination Test Cases

Conditions	**Observations**
DS_ServiceAttachRequest was valid	DS_ServiceAttachConfirm has response OK
DS_ServiceAttachRequest was not valid	DS_ServiceAttachConfirm has response NOT OK

6.3.2.1.2.2.3 Test Case 3 - Detaching a Service (originating side)

Test Purpose:

Verify that a Service Detach procedure is correctly performed at the originating side.

Test Preamble:

bb) The UT requests through a DA_ServiceDetach(IN) to detach a service

cc) The service was existing and operating at the LT, over an existing session

Test Procedure:

39 UT passes a DA_ServiceDetach(IN) to the IUT

40 IUT sends a 'DS_ServiceDetachRequest' message

41 LT responds with a 'DS_ServiceDetachConfirm' message

42 IUT passes a DA_ServiceDetach(OUT) back to the UT

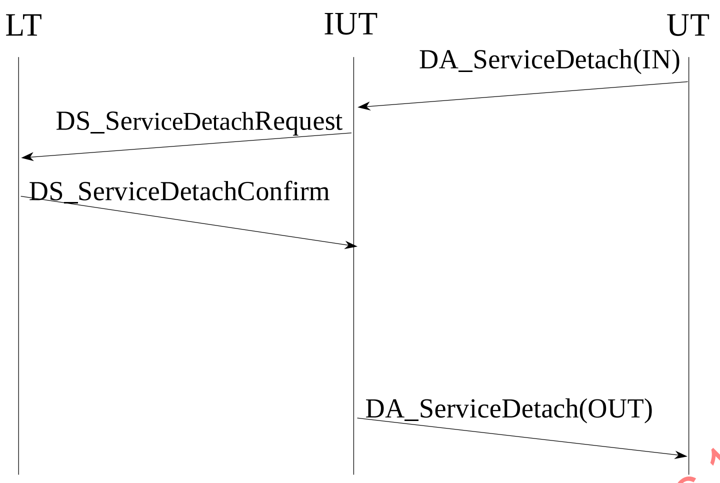

Figure 6-9 — Service Detach Origination side

Test Verdict:

Pass the test if the response to UT is:

Table 6-15 — Service Detach Origination Test Cases

Conditions	Observations
DA_ServiceDetach(IN) was valid DS_ServiceDetachConfirm had response OK	DS_ServiceDetachRequest has been originated consistently DA_ServiceDetach(OUT) has been generated consistently with response OK
DA_ServiceDetach(IN) was valid DS_ServiceDetachConfirm had response not OK	DS_ServiceDetachRequest has been originated consistently DA_ServiceDetach(OUT) has been generated consistently with response not OK
DA_ServiceDetach(IN) was not valid	DS_ServiceDetachRequest has not been originated DA_ServiceDetach(OUT) has been generated consistently with response not OK

6.3.2.1.2.2.4 Test Case 4 - Detaching a Service (destination side)

Test Purpose:

Verify that a Service Detach procedure is correctly performed at the destination side.

Test Preamble:

dd) The service was existing and operating between SUT and LT

Test Procedure:

43 LT sends a 'DS_ServiceDetachRequest' message.

44 IUT responds with a 'DS_ServiceDetachConfirm' message

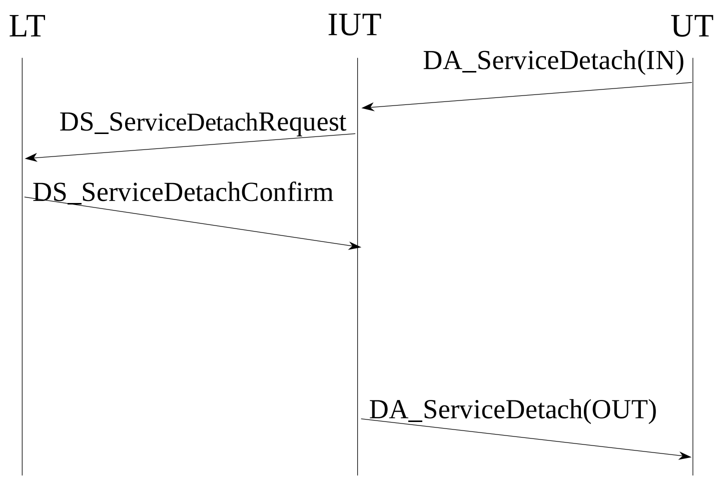

Figure 6-10 — Service Detach Destination side

Test Verdict:

Pass the test if the response to LT is:

Table 6-16 — Service Detach Destination Test Cases

Conditions	**Observations**
DS_ServiceDetachRequest was valid	DS_ServiceDetachConfirm has response OK
DS_ServiceDetachRequest was not valid	DS_ServiceDetachConfirm has response NOT OK

6.3.2.1.2.3 DMIF Transmux primitives

Table 6-17 — DMIF Transmux Primitives Test Cases

Test Case #	**Test Case Names**	**Reference to ISO/IEC 14496-6**
1	Setting up a Transmux (originating side)	12.1.2.9, 12.1.2.10
2	Setting up a Transmux (destination side)	12.1.2.9, 12.1.2.10
3	Releasing a Transmux (originating side)	12.1.2.11, 12.1.2.12
4	Releasing a Transmux (destination side)	12.1.2.11, 12.1.2.12

6.3.2.1.2.3.1 Test Case 1 - Setting up a Transmux (originating side)

Test Purpose:

Verify that a Transmux Setup procedure is correctly performed at the originating side. This includes also a Channel Add procedure.

Test Preamble:

ee) The UT requests through a DA_ChannelAdd(IN) to add a new channel to a service

ff) The service is located at the LT, and is already existing and operating

Test Procedure:

45 UT passes a DA_ChannelAdd(IN) to the IUT

46 IUT sends a 'DS_TransmuxSetupRequest' message

47 LT responds with a 'DS_TransmuxSetupConfirm' message

48 IUT sends a 'DS_ChannelAddedRequest' message

49 LT responds with a 'DS_ChannelAddedConfirm' message

50 IUT passes a DA_ChannelAdd(OUT) back to the UT

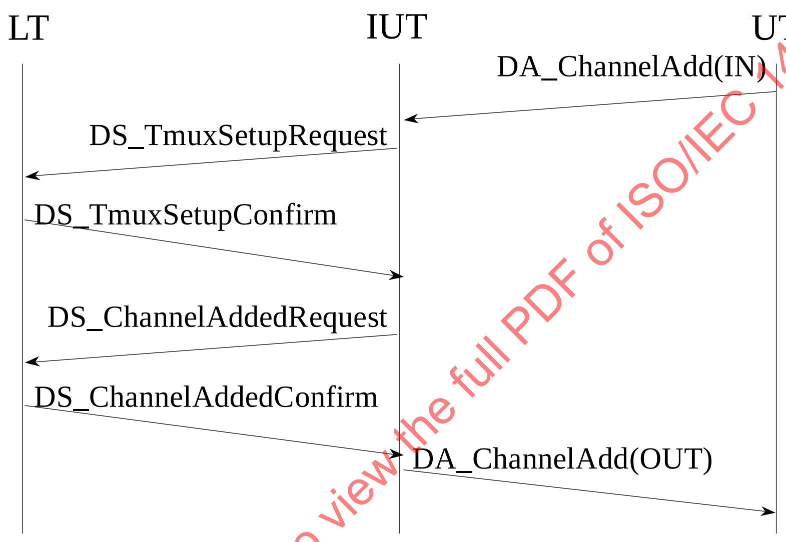

Figure 6-11 — Transmux Setup Origination side

Test Verdict:

Pass the test if the response to UT is:

Table 6-18 — Transmux Setup Origination Test Cases

Conditions	Observations
DA_ChannelAdd(IN) was valid DS_TransmuxSetupConfirm had response OK DS_ChannelAddedConfirm had response OK	DS_TransmuxSetupRequest has been originated consistently DS_ChannelAddedRequest has been originated consistently DA_ChannelAdd(OUT) has been generated consistently with response OK
DA_ChannelAdd(IN) was valid DS_TransmuxSetupConfirm had response OK DS_ChannelAddedConfirm had response not OK	DS_TransmuxSetupRequest has been originated consistently DS_ChannelAddedRequest has been originated consistently DA_ChannelAdd(OUT) has been generated consistently with response not OK and a Transmux Release procedure has been initiated
DA_ChannelAdd(IN) was valid DS_TransmuxSetupConfirm had response not OK	DS_TransmuxSetupRequest has been originated consistently DS_ChannelAddedRequest has not been originated DA_ChannelAdd(OUT) has been generated consistently with response not OK
DA_ChannelAdd(IN) was not valid	DS_TransmuxSetupRequest has not been originated DA_ChannelAdd(OUT) has been generated consistently with response not OK

6.3.2.1.2.3.2 Test Case 2 - Setting up a Transmux (destination side)

Test Purpose:

Verify that a Transmux Setup procedure is correctly performed at the destination side.

Test Preamble:

gg) A service is existing and operating between SUT and LT

Test Procedure:

51 LT sends a 'DS_TransmuxSetupRequest' message.

52 IUT responds with a 'DS_TransmuxSetupConfirm' message

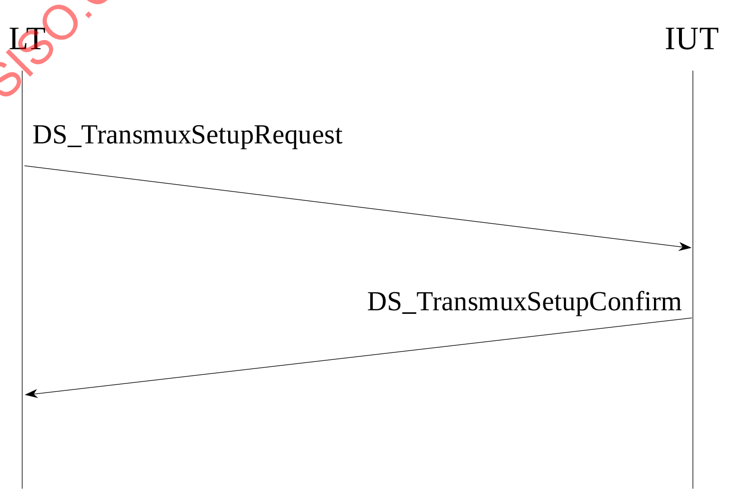

Figure 6-12 — Transmux Setup Destination side

Test Verdict:

Pass the test if the response to LT is:

Table 6-19 — Transmux Setup Destination Test Cases

Conditions	Observations
DS_TransmuxSetupRequest was valid	DS_TransmuxSetupConfirm has response OK
DS_TransmuxSetupRequest was not valid	DS_TransmuxSetupConfirm has response NOT OK

6.3.2.1.2.3.3 Test Case 3 - Releasing a Transmux (originating side)

Test Purpose:

Verify that a Transmux Release procedure is correctly performed at the originating side. This includes also a Service Detach procedure.

Test Preamble:

hh) The UT requests through a DA_ChannelDelete(IN) to delete a channel

ii) The channel was existing and operating between the SUT and the LT, over an existing Transmux

Test Procedure:

53 UT passes a DA_ChannelDelete(IN) to the IUT

54 IUT sends a 'DS_ChannelDeleteRequest' message

55 LT responds with a 'DS_ChannelDeleteConfirm' message

56 IUT sends a 'DS_TransmuxReleaseRequest' message

57 LT responds with a 'DS_TransmuxReleaseConfirm' message

58 IUT passes a DA_ChannelDelete(OUT) back to the UT

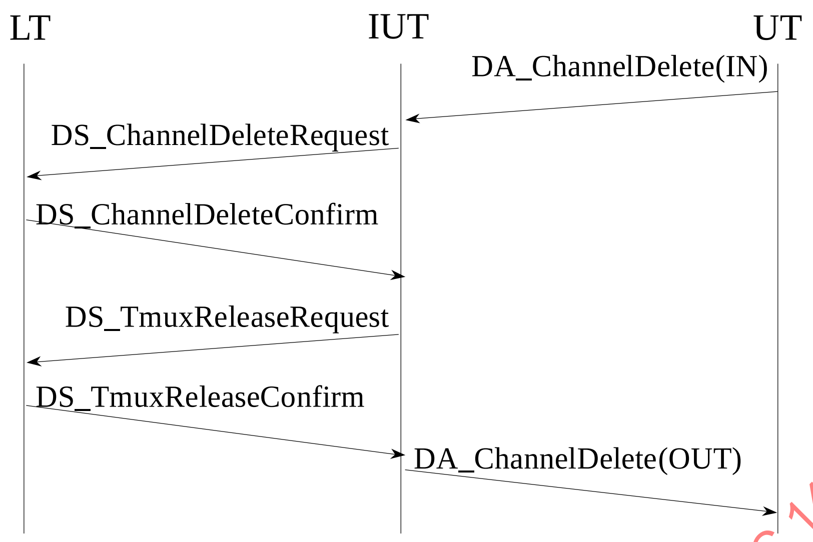

Figure 6-13 — Transmux Release Origination side

Test Verdict:

Pass the test if the response to UT is:

Table 6-20 — Transmux Release Origination Test Cases

Conditions	**Observations**
DA_ChannelDelete(IN) was valid DS_ChannelDeleteConfirm had response OK DS_TransmuxReleaseConfirm had response OK	DS_ChannelDeleteRequest has been originated consistently DS_TransmuxReleaseRequest has been originated consistently DA_ChannelDelete(OUT) has been generated consistently with response OK
DA_ChannelDelete(IN) was valid DS_ChannelDeleteConfirm had response OK DS_TransmuxReleaseConfirm had response not OK	DS_ChannelDeleteRequest has been originated consistently DS_TransmuxReleaseRequest has been originated consistently DA_ChannelDelete(OUT) has been generated consistently with response OK
DA_ChannelDelete(IN) was valid DS_ChannelDeleteConfirm had response not OK	DS_ChannelDeleteRequest has been originated consistently DS_TransmuxReleaseRequest has not been originated DA_ChannelDelete(OUT) has been generated consistently with response not OK
DA_ChannelDelete(IN) was not valid	DS_ChannelDeleteRequest has not been originated DA_ChannelDelete(OUT) has been generated consistently with response not OK

6.3.2.1.2.3.4 Test Case 4 - Releasing a Transmux (destination side)

Test Purpose:

Verify that a Transmux Release procedure is correctly performed at the destination side.

Test Preamble:

jj) The Transmux was existing and operating between SUT and LT

Test Procedure:

59 LT sends a 'DS_TransmuxReleaseRequest' message.

60 IUT responds with a 'DS_TransmuxReleaseConfirm' message

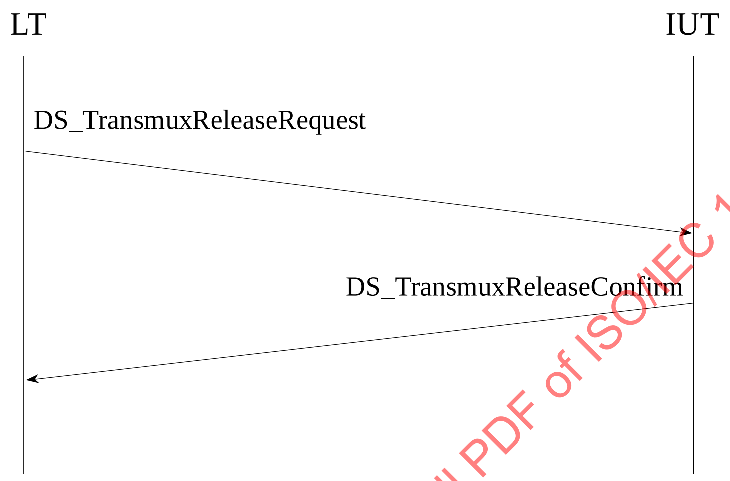

Figure 6-14 — Transmux Release Destination side

Test Verdict:

Pass the test if the response to LT is:

Table 6-21 — Transmux Release Destination Test Cases

Conditions	Observations
DS_TransmuxReleaseRequest was valid	DS_TransmuxReleaseConfirm has response OK
DS_TransmuxReleaseRequest was not valid	DS_TransmuxReleaseConfirm has response NOT OK

6.3.2.1.2.4 DMIF Channel Primitives

Table 6-22 — DMIF DAI Service Primitives Test Cases

Test Case #	Test Case Names	Reference to ISO/IEC 14496-6
1	Adding a Channel (originating side)	12.1.2.13, 12.1.2.14, 12.1.2.15, 12.1.2.16
2	Adding a Channel (destination side)	12.1.2.13, 12.1.2.14, 12.1.2.15, 12.1.2.16
3	Deleting a Channel (originating side)	12.1.2.17, 12.1.2.18
4	Deleting a Channel (destination side)	12.1.2.17, 12.1.2.18

6.3.2.1.2.4.1 Test Case 1 - Adding a Channel (originating side)

Test Purpose:

Verify that a Channel Add procedure is correctly performed at the originating side.

Test Preamble:

kk) The UT requests through a DA_ChannelAdd(IN) to add a new channel

ll) A Transmux with the LT is already existing and operating

Test Procedure:

61 UT passes a DA_ChannelAdd(IN) to the IUT

62 IUT sends a 'DS_ChannelAddRequest' message

63 LT responds with a 'DS_ChannelAddConfirm' message

64 IUT passes a DA_ChannelAdd(OUT) back to the UT

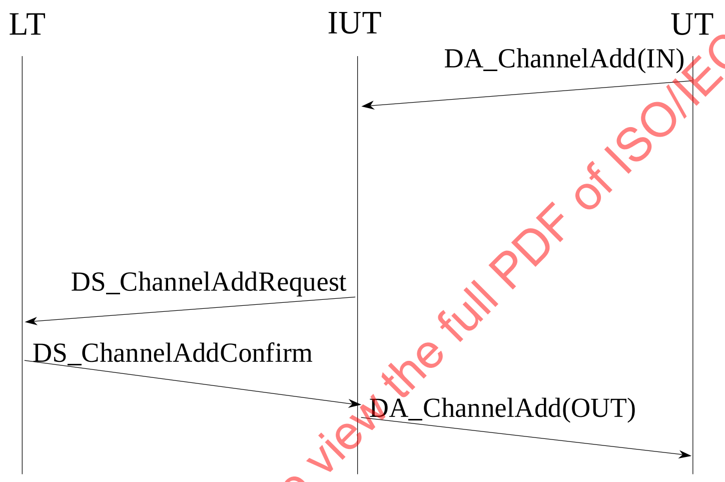

Figure 6-15 — Channel Add Origination side

Test Verdict:

Pass the test if the response to UT is:

Table 6-23 — Channel Add Origination Test Cases

Conditions	**Observations**
DA_ChannelAdd(IN) was valid DS_ChannelAddConfirm had response OK	DS_ChannelAddRequest has been originated consistently DA_ChannelAdd(OUT) has been generated consistently with response OK
DA_ChannelAdd(IN) was valid DS_ChannelAddConfirm had response not OK	DS_ChannelAddRequest has been originated consistently DA_ChannelAdd(OUT) has been generated consistently with response not OK
DA_ChannelAdd(IN) was not valid	DS_ChannelAddRequest has not been originated DA_ChannelAdd(OUT) has been generated consistently with response not OK

6.3.2.1.2.4.2 Test Case 2 - Adding a Channel (destination side)

Test Purpose:

Verify that a Channel Add procedure is correctly performed at the destination side.

Test Preamble:

mm) A Transmux is existing and operating between SUT and LT

Test Procedure:

65 LT sends a 'DS_ChannelAddRequest' message.

66 IUT responds with a 'DS_ChannelAddConfirm' message

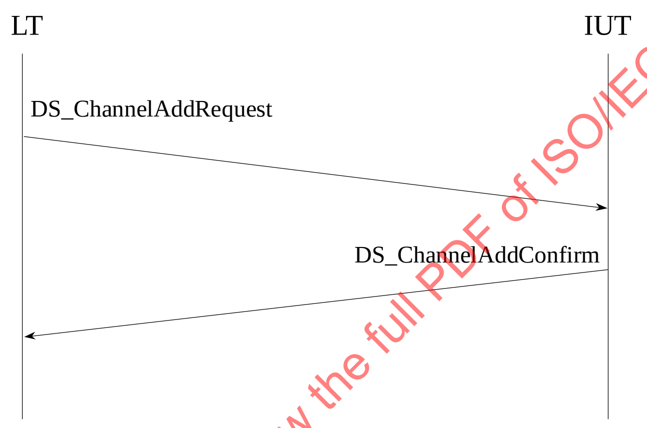

Figure 6-16 — Service Attach Destination side

Test Verdict:

Pass the test if the response to LT is:

Table 6-24 — Service Attach Destination Test Cases

Conditions	Observations
DS_ChannelAddRequest was valid	DS_ChannelAddConfirm has response OK
DS_ChannelAddRequest was not valid	DS_ChannelAddConfirm has response NOT OK

6.3.2.1.2.4.3 Test Case 3 - Deleting a Channel (originating side)

Test Purpose:

Verify that a Service Detach procedure is correctly performed at the originating side.

Test Preamble:

nn) The UT requests through a DA_ChannelDelete(IN) to delete a channel

oo) The channel was existing and operating at the LT, over an existing Transmux

Test Procedure:

67 UT passes a DA_ChannelDelete(IN) to the IUT

68 IUT sends a ‘DS_ChannelDeleteRequest’ message

69 LT responds with a ‘DS_ChannelDeleteConfirm’ message

70 IUT passes a DA_ChannelDelete(OUT) back to the UT

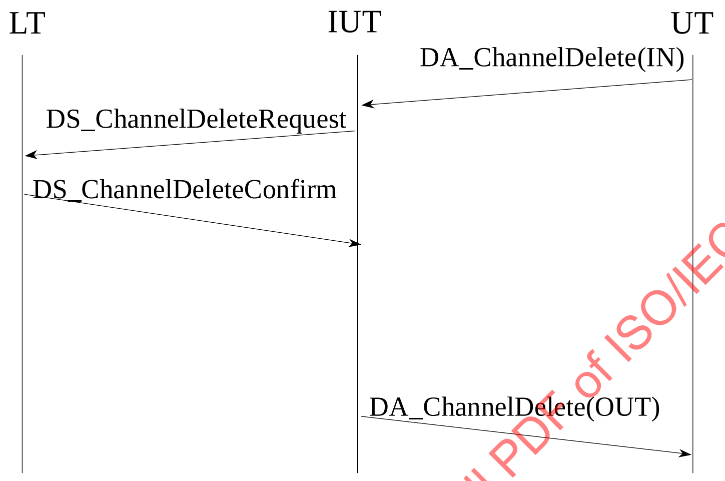

Figure 6-17 — Channel Delete Origination side

Test Verdict:

Pass the test if the response to UT is:

Table 6-25 — Channel Delete Origination Test Cases

Conditions	**Observations**
DA_ChannelDelete(IN) was valid DS_ChannelDeleteConfirm had response OK	DS_ChannelDeleteRequest has been originated consistently DA_ChannelDelete(OUT) has been generated consistently with response OK
DA_ChannelDelete(IN) was valid DS_ChannelDeleteConfirm had response not OK	DS_ChannelDeleteRequest has been originated consistently DA_ChannelDelete(OUT) has been generated consistently with response not OK
DA_ChannelDelete(IN) was not valid	DS_ChannelDeleteRequest has not been originated DA_ChannelDelete(OUT) has been generated consistently with response not OK

6.3.2.1.2.4.4 Test Case 4 - Deleting a Channel (destination side)

Test Purpose:

Verify that a Channel Delete procedure is correctly performed at the destination side.

Test Preamble:

pp) The channel was existing and operating between SUT and LT

Test Procedure:

71 LT sends a ‘DS_ChannelDeleteRequest’ message.

72 IUT responds with a ‘DS_ChannelDeleteConfirm’ message

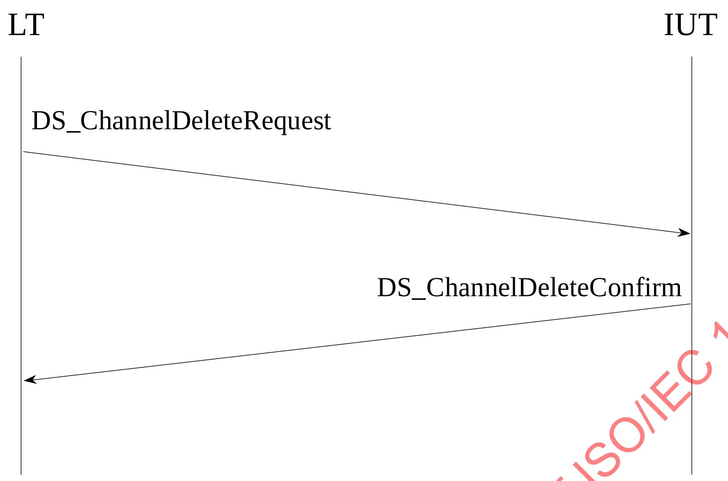

Figure 6-18 — Channel Delete Destination side

Test Verdict:

Pass the test if the response to LT is:

Table 6-26 — Channel Delete Destination Test Cases

Conditions	**Observations**
DS_ChannelDeleteRequest was valid	DS_ChannelDeleteConfirm has response OK
DS_ChannelDeleteRequest was not valid	DS_ChannelDeleteConfirm has response NOT OK

6.3.2.1.3 Test Coverage

6.3.2.2 Specific ATS for DS over ATM

6.3.2.2.1 Test Environment

The test environment described in subclause 6.3.2.1 is valid.

6.3.2.2.2 Test Cases

6.3.2.2.2.1 DMIF Session primitives

Table 6-27 — DMIF Session Primitives Test Cases

Test Case #	Test Case Names	Reference to ISO/IEC 14496-6
1	Setting up a Session (originating side)	12.1.2.1, 12.1.2.2, 12.4.5.1
2	Setting up a Session (destination side)	12.1.2.1, 12.1.2.2, 12.4.5.1
3	Releasing a Session (originating side)	12.1.2.3, 12.1.2.4, 12.4.5.2
4	Releasing a Session (destination side)	12.1.2.3, 12.1.2.4, 12.4.5.2

6.3.2.2.2.1.1 Test Case 1 - Setting up a Session (originating side)

Test Purpose:

Verify that a Session Setup procedure is correctly performed at the originating side. This includes also a Service Attach procedure.

Test Preamble:

qq) The UT requests through a DA_ServiceAttach(IN) to create a new service

rr) The service is located at the LT

ss) The SUT knows how to contact the LT

Test Procedure:

73 UT passes a DA_ServiceAttach(IN) to the IUT

74 IUT sends a ‘Q.2931 SETUP’ with BHLI message

75 LT responds with a ‘Q2931 CONNECT’ message

76 IUT sends a ‘DS_ServiceAttachRequest’ message

77 LT responds with a ‘DS_ServiceAttachConfirm’ message

78 IUT passes a DA_ServiceAttach(OUT) back to the UT

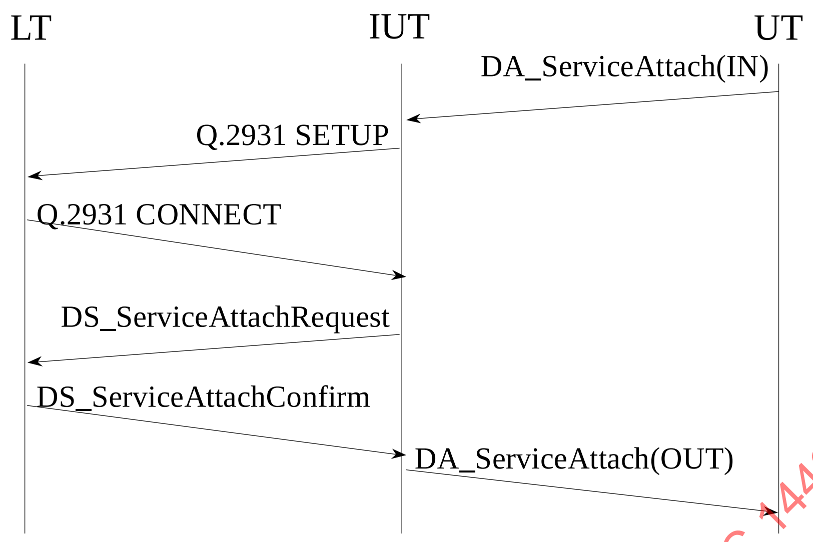

Figure 6-19 — Session Setup with Q.2931 Origination side

Test Verdict:

Pass the test if the response to UT is:

Table 6-28 — Session Setup with Q.2931 Origination Test Cases

Conditions	**Observations**
DA_ServiceAttach(IN) was valid Q.2931 CONNECT had response OK DS_ServiceAttachConfirm had response OK	Q.2931 SETUP has been originated consistently DS_ServiceAttachRequest has been originated consistently DA_ServiceAttach(OUT) has been generated consistently with response OK
DA_ServiceAttach(IN) was valid Q.2931 CONNECT had response OK DS_ServiceAttachConfirm had response not OK	Q.2931 SETUP has been originated consistently DS_ServiceAttachRequest has been originated consistently DA_ServiceAttach(OUT) has been generated consistently with response not OK and a Session Release procedure has been initiated
DA_ServiceAttach(IN) was valid Q.2931 CONNECT had response not OK	Q.2931 SETUP has been originated consistently DS_ServiceAttachRequest has not been originated DA_ServiceAttach(OUT) has been generated consistently with response not OK
DA_ServiceAttach(IN) was not valid	Q.2931 SETUP has not been originated DA_ServiceAttach(OUT) has been generated consistently with response not OK

6.3.2.2.2.1.2 Test Case 2 - Setting up a Session (destination side)

Test Purpose:

Verify that a Session Setup procedure is correctly performed at the destination side.

Test Preamble:

tt) The LT knows how to contact the SUT

Test Procedure:

79 LT sends a ‘Q.2931 SETUP’ message.

80 IUT responds with a ‘Q.2931 CONNECT’ message

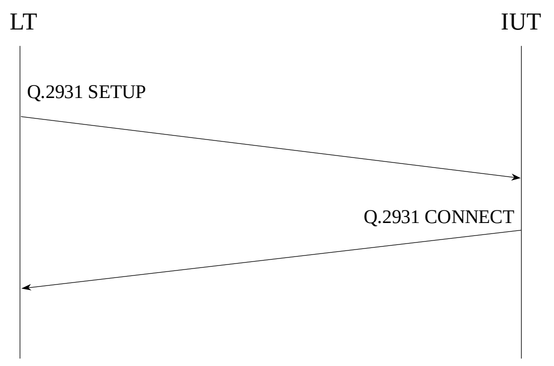

Figure 6-20 — Session Setup with Q.2931 Destination side

Test Verdict:

Pass the test if the response to LT is:

Table 6-29 — Session Setup with Q.2931 Destination Test Cases

Conditions	**Observations**
Q.2931 SETUP was valid	Q.2931 CONNECT has response OK
Q.2931 SETUP was not valid	Q.2931 CONNECT has response NOT OK

6.3.2.2.2.1.3 Test Case 3 - Releasing a Session (originating side)

Test Purpose:

Verify that a Session Release procedure is correctly performed at the originating side. This includes also a Service Detach procedure.

Test Preamble:

uu) The UT requests through a DA_ServiceDetach(IN) to detach a service

vv) The service was existing and operating at the LT, over an existing session

Test Procedure:

81 UT passes a DA_ServiceDetach(IN) to the IUT

82 IUT sends a ‘DS_ServiceDetachRequest’ message

83 LT responds with a ‘DS_ServiceDetachConfirm’ message

84 IUT sends a ‘Q.2931 RELEASE’ message

85 LT responds with a ‘Q.2931 RELEASE COMPLETE’ message

86 IUT passes a DA_ServiceDetach(OUT) back to the UT

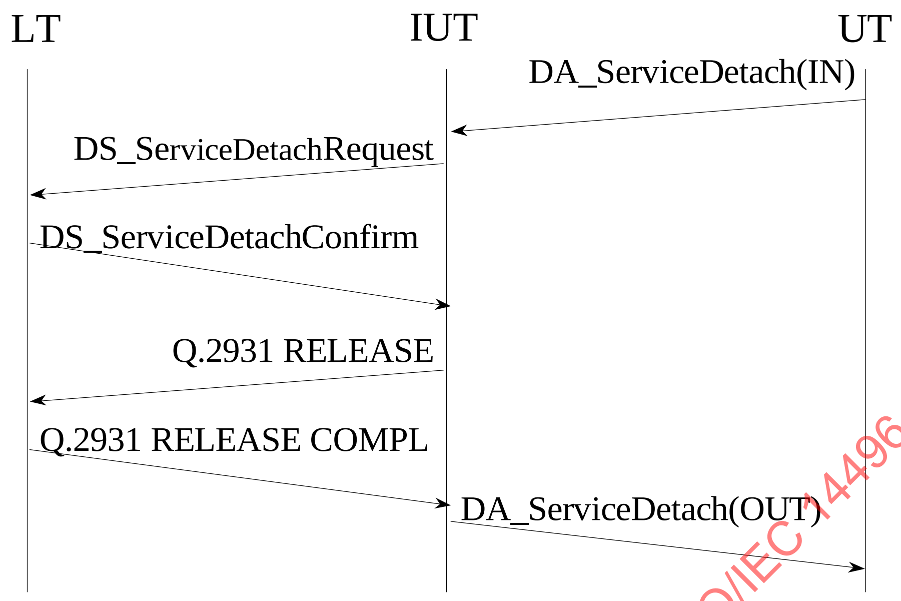

Figure 6-21 — Session Release with Q.2931 Origination side

Test Verdict:

Pass the test if the response to UT is:

Table 6-30 — Session Release with Q.2931 Origination Test Cases

Conditions	**Observations**
DA_ServiceDetach(IN) was valid DS_ServiceDetachConfirm had response OK Q.2931 RELEASE COMPLETE had response OK	DS_ServiceDetachRequest has been originated consistently Q.2931 RELEASE has been originated consistently DA_ServiceDetach(OUT) has been generated consistently with response OK
DA_ServiceDetach(IN) was valid DS_ServiceDetachConfirm had response OK Q.2931 RELEASE COMPLETE had response not OK	DS_ServiceDetachRequest has been originated consistently Q.2931 RELEASE has been originated consistently DA_ServiceDetach(OUT) has been generated consistently with response OK
DA_ServiceDetach(IN) was valid DS_ServiceDetachConfirm had response not OK	DS_ServiceDetachRequest has been originated consistently Q.2931 RELEASE has not been originated DA_ServiceDetach(OUT) has been generated consistently with response not OK
DA_ServiceDetach(IN) was not valid	DS_ServiceDetachRequest has not been originated DA_ServiceDetach(OUT) has been generated consistently with response not OK

6.3.2.2.2.1.4 Test Case 4 - Releasing a Session (destination side)

Test Purpose:

Verify that a Session Release procedure is correctly performed at the destination side.

Test Preamble:

ww) The session was existing and operating between SUT and LT

Test Procedure:

87 LT sends a 'Q.2931 RELEASE' message.

88 IUT responds with a 'Q.2931 RELEASE COMPLETE' message

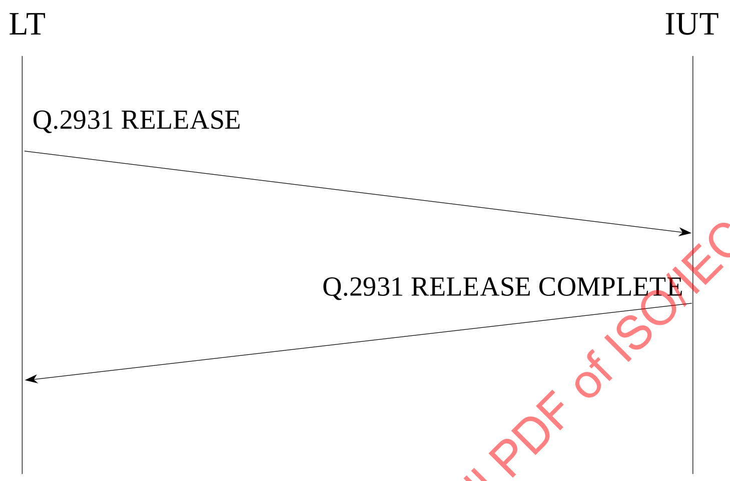

Figure 6-22 — Session Release with Q.2931 Destination side

Test Verdict:

Pass the test if the response to LT is:

Table 6-31 — Session Release with Q.2931 Destination Test Cases

Conditions	Observations
Q.2931 RELEASE was valid	Q.2931 RELEASE COMPLETE has response OK
Q.2931 RELEASE was not valid	Q.2931 RELEASE COMPLETE has response NOT OK

6.3.2.2.2.2 DMIF Service Primitives

The test cases described in subclause 6.3.2.1.2.2 are valid.

6.3.2.2.2.3 DMIF Transmux primitives

Table 6-32 — DMIF Transmux Primitives Test Cases

Test Case #	Test Case Names	Reference to ISO/IEC 14496-6
1	Setting up a Transmux (originating side)	12.1.2.9, 12.1.2.10, 12.4.5.3
2	Setting up a Transmux (destination side)	12.1.2.9, 12.1.2.10, 12.4.5.3
3	Releasing a Transmux (originating side)	12.1.2.11, 12.1.2.12, 12.4.5.4
4	Releasing a Transmux (destination side)	12.1.2.11, 12.1.2.12, 12.4.5.4

6.3.2.2.2.3.1 Test Case 1 - Setting up a Transmux (originating side)

Test Purpose:

Verify that a Transmux Setup procedure is correctly performed at the originating side. This includes also a Channel Add procedure.

Test Preamble:

xx) The UT requests through a DA_ChannelAdd(IN) to add a new channel to a service

yy) The service is located at the LT, and is already existing and operating

Test Procedure:

89 UT passes a DA_ChannelAdd(IN) to the IUT

90 IUT sends a ‘Q.2931 SETUP’ message

91 LT responds with a ‘Q.2931 CONNECT’ message

92 IUT sends a ‘DS_ChannelAddedRequest’ message

93 LT responds with a ‘DS_ChannelAddedConfirm’ message

94 IUT passes a DA_ChannelAdd(OUT) back to the UT

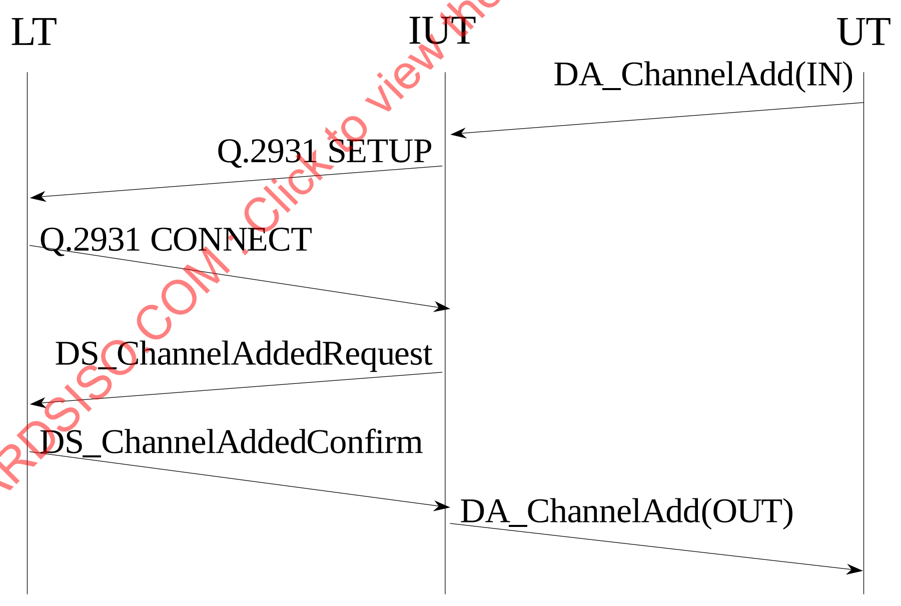

Figure 6-23 — Transmux Setup with Q.2931 Origination side

Test Verdict:

Pass the test if the response to UT is:

Table 6-33 — Transmux Setup with Q.2931 Origination Test Cases

Conditions	Observations
DA_ChannelAdd(IN) was valid Q.2931 CONNECT had response OK DS_ChannelAddedConfirm had response OK	Q.2931 SETUP has been originated consistently DS_ChannelAddedRequest has been originated consistently DA_ChannelAdd(OUT) has been generated consistently with response OK
DA_ChannelAdd(IN) was valid Q.2931 CONNECT had response OK DS_ChannelAddedConfirm had response not OK	Q.2931 SETUP has been originated consistently DS_ChannelAddedRequest has been originated consistently DA_ChannelAdd(OUT) has been generated consistently with response not OK and a Transmux Release procedure has been initiated
DA_ChannelAdd(IN) was valid Q.2931 CONNECT had response not OK	Q.2931 SETUP has been originated consistently DS_ChannelAddedRequest has not been originated DA_ChannelAdd(OUT) has been generated consistently with response not OK
DA_ChannelAdd(IN) was not valid	Q.2931 SETUP has not been originated DA_ChannelAdd(OUT) has been generated consistently with response not OK

6.3.2.2.2.3.2 Test Case 2 - Setting up a Transmux (destination side)

Test Purpose:

Verify that a Transmux Setup procedure is correctly performed at the destination side.

Test Preamble:

zz) A service is existing and operating between SUT and LT

Test Procedure:

95 LT sends a 'Q.2931 SETUP' message.

96 IUT responds with a 'Q.2931 CONNECT' message

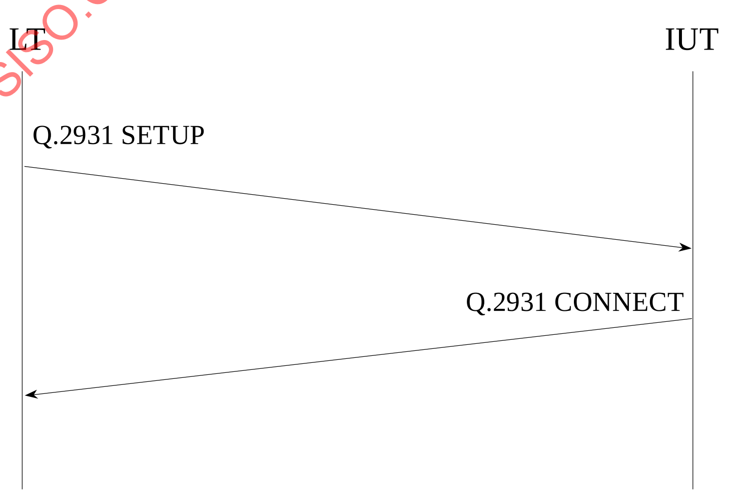

Figure 6-24 — Transmux Setup with Q.2931 Destination side

Test Verdict:

Pass the test if the response to LT is:

Table 6-34 — Transmux Setup with Q.2931 Destination Test Cases

Conditions	**Observations**
Q.2931 SETUP was valid	Q.2931 CONNECT has response OK
Q.2931 SETUP was not valid	Q.2931 CONNECT has response NOT OK

6.3.2.2.2.3.3 Test Case 3 - Releasing a Transmux (originating side)

Test Purpose:

Verify that a Transmux Release procedure is correctly performed at the originating side. This includes also a Service Detach procedure.

Test Preamble:

aaa) The UT requests through a DA_ChannelDelete(IN) to delete a channel

bbb) The channel was existing and operating between the SUT and the LT, over an existing Transmux

Test Procedure:

97 UT passes a DA_ChannelDelete(IN) to the IUT

98 IUT sends a ‘DS_ChannelDeleteRequest’ message

99 LT responds with a ‘DS_ChannelDeleteConfirm’ message

100 IUT sends a ‘Q.2931 RELEASE’ message

101 LT responds with a ‘Q.2931 RELEASE COMPLETE’ message

102 IUT passes a DA_ChannelDelete(OUT) back to the UT

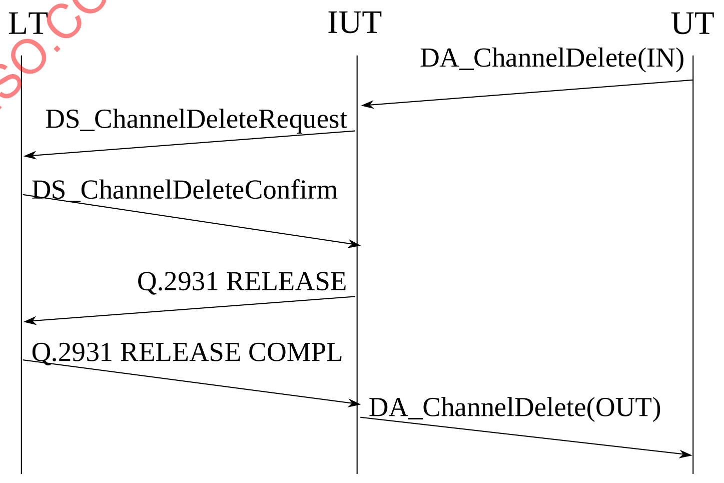

Figure 6-25 — Transmux Release with Q.2931 Origination side

Test Verdict:

Pass the test if the response to UT is:

Table 6-35 — Transmux Release with Q.2931 Origination Test Cases

Conditions	Observations
DA_ChannelDelete(IN) was valid DS_ChannelDeleteConfirm had response OK Q.2931 RELEASE COMPLETE had response OK	DS_ChannelDeleteRequest has been originated consistently Q.2931 RELEASE has been originated consistently DA_ChannelDelete(OUT) has been generated consistently with response OK
DA_ChannelDelete(IN) was valid DS_ChannelDeleteConfirm had response OK Q.2931 RELEASE COMPLETE had response not OK	DS_ChannelDeleteRequest has been originated consistently Q.2931 RELEASE has been originated consistently DA_ChannelDelete(OUT) has been generated consistently with response OK
DA_ChannelDelete(IN) was valid DS_ChannelDeleteConfirm had response not OK	DS_ChannelDeleteRequest has been originated consistently Q.2931 RELEASE has not been originated DA_ChannelDelete(OUT) has been generated consistently with response not OK
DA_ChannelDelete(IN) was not valid	DS_ChannelDeleteRequest has not been originated DA_ChannelDelete(OUT) has been generated consistently with response not OK

6.3.2.2.2.3.4 Test Case 4 - Releasing a Transmux (destination side)

Test Purpose:

Verify that a Transmux Release procedure is correctly performed at the destination side.

Test Preamble:

ccc) The Transmux was existing and operating between SUT and LT

Test Procedure:

103 LT sends a 'Q.2931 RELEASE' message.

104 IUT responds with a 'Q.2931 RELEASE COMPLETE' message

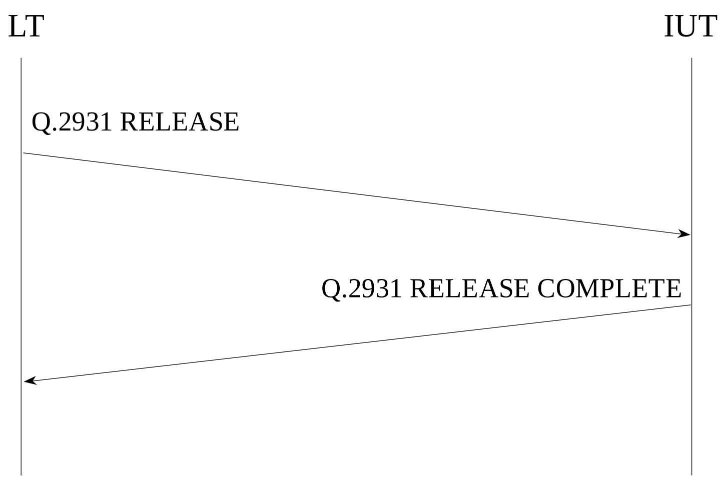

Figure 6-26 — Transmux Release with Q.2931 Destination side

Test Verdict:

Pass the test if the response to LT is:

Table 6-36 — Transmux Release with Q.2931 Destination Test Cases

Conditions	**Observations**
Q.2931 RELEASE was valid	Q.2931 RELEASE COMPLETE has response OK
Q.2931 RELEASE was not valid	Q.2931 RELEASE COMPLETE has response NOT OK

6.3.2.2.2.4 DMIF Channel Primitives

The test cases described in subclause 6.3.2.1.2.4 are valid.

7 SNHC

7.1 Introduction

7.1.1 Purpose & Scope

The purpose of this clause 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 functionality of Face Animation and 2D Animated Mesh 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 functionality specified in ISO/IEC 14496 can depend on the unified application of related features in ISO/IEC 14496-1. In particular, SNHC functionality can be deployed with elements of ISO/IEC 14496-1 including elementary streams, scene description and composition supported by the Binary Interchange Format for Scenes (BIFS).

An example is the downloading of Face Definition Parameters to describe a specific face model in preparation for subsequent animation of that model by Face Animation Parameters. Another example is the application of a 2D Animated Mesh to warp an associated texture where composition functionality of ISO/IEC 14496-1 integrate the 2D mesh and texture. 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.

7.1.2 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, Face Animation can be used in a broadcast application without any terminal utilisation 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 functionality and performance without connection to Systems.

Face Animation 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 customised model. Then testing should be conducted with the specified decoder and BIFS functionality. 2D Animated Mesh can also be accomplished with elements of both the Systems and Visual parts and shall be tested accordingly.

7.1.3 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 functionality and modes of decoding to produce expected results,

   2. Exercising the full range of bitstream rates, frame rates, skip frame, quantisation 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 utilising 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.

7.2 Conformance Points

7.2.1 Covered Functionality

7.2.1.1 Face Animation

Face Animation 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 functionality 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 Face Animation capability can be used without or with other features of Face Animation supported in Systems BIFS, again depending on the application.

The Systems BIFS features supporting Face Animation include:

1. the Face Definition Parameters (FDP) in BIFS (model data downloadable to configure a baseline face model pre-stored in the terminal into a particular face before FAP decoding, or to install a specific face model at the beginning of a session along with the information about how to animate it),

2. the Face Animation Table (FAT) within FDPs (downloadable functional mapping from incoming FAPs to feature control points in the face mesh that provides piecewise linear weightings of incoming FAPs for controlling facial movements), and

3. the Face Interpolation Transform (FIT) in BIFS (downloadable definition of an optional mapping of incoming FAPs into a total set of FAPs before their application to feature points, through weighted rational polynomial functions invoked by conditional evaluation of a Face Interpolation Graph) for complex cross-coupling of FAPs to link their effects into varied face models or to interpolate missing FAPs in the stream into required FAPs in the terminal).

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 customise and animate the model, and/or the display of specific characteristics in the final face rendering of non-normative models downloaded under BIFS.

7.2.1.2 2D Mesh Animation

2D Mesh Animation in ISO/IEC 14496 includes the specification of highly efficient coding of the geometry and motion of a 2D triangular mesh, which can drive an animation of a 2D visual object. This mesh can be either a uniform array of triangles or an object-based mesh of triangles.

The mesh geometry for an animation sequence is coded at the start of the animation, or, more generally, at intra-frames. These initialisation frames are followed by a series of inter-frames that advance the 2D locations of all mesh vertices with motion vector coding. Such a mesh sequence may originate from the use of appropriate encoder techniques to track feature movement within natural video frames or may be generated synthetically, and can be used to manipulate a texture map for special animation effects.

Baseline conformance testing of 2D Mesh Animation includes demonstration of mesh decoder conformance to specified profile and level requirements on mesh node complexities and motion vector throughputs, independent of any integration of the decoder in a complete ISO/IEC 14496 System. The decoder must demonstrate the proper output of a mesh sequence with either uniform or object-based topology. A complete system for 2D Mesh Animation requires coupling of the decoded mesh sequence with decoded still texture or video object data for rendering.

7.2.2 Description/References on Conformance Definitions

7.2.2.1 Face Animation

7.2.2.1.1 Introduction

The required decoder functionality to be achieved in testing is specified in the following detailed clauses of ISO/IEC 14496-1 and ISO/IEC 14496-2:

The Face Animation 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 specification is found in ISO/IEC 14496-2, and FDP specification in ISO/IEC 14496-1. However, this is not a strict rule. For an overview of FAPs and their interpretation, read subclauses 6.1.5.2, 6.1.5.3, and 6.1.5.4 as well as Table C.1 of ISO/IEC 14496-2. The viseme parameter is documented in subclause 7.12.3 and Table C.5 of ISO/IEC 14496-2. The expression parameter is documented in subclause 7.12.4 and Table C.3 of ISO/IEC 14496-2. FAP bitstream syntax is found in subclause 6.2.10, semantics in subclause 6.3.10, and subclause 7.12 of ISO/IEC 14496-2 explains in more detail the FAP decoding process. FAP masking and interpolation is explained in subclauses 6.3.11.1, 7.12.1.1, and 7.12.5 of ISO/IEC 14496-2. The FIT interpolation scheme is documented in subclause 7.2.5.3.2.4 of ISO/IEC 14496-1. The FDPs and their interpretation are documented in subclause 7.2.5.3.2.6 of ISO/IEC 14496-1. In particular, the FDP feature points are documented in Annex C of ISO/IEC 14496-2.

7.2.2.1.2 Related Systems BIFS Nodes, Elementary Streams

Table 7-1 — Related Systems BIFS Nodes, Elementary Streams

N°	Node or Concept	Functionality	Test sequence	Hints
1	Face	FAP animation	FaceAndCube	
2	Face	FDP Feature Points calibration		
3	Face	FDP face model download, FDP, FaceDefTransform, FaceDefMesh, FaceDefTable	FATFace	
4	Face	FIT	see subclause 1.1.1.1.1.3	
5	Face	TTS	FaceTTS	

7.2.2.1.3 Profile and Level Definitions

The profile and levels that Face Animation are defined in clause 9. These profiles are:

1. Simple Face Animation Profile,

2. Basic Animated Texture Profile

3. Hybrid 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”.

7.2.2.2 2D Mesh Animation

7.2.2.2.1 Introduction

Figure 7-1 illustrates a model for decoding, composition and rendering of a scene with 2D mesh animation. A BIFS bitstream with the scene description is decoded by the BIFS decoder, as is specified in ISO/IEC 14496-1. The scene description does not describe any mesh geometry or image content, rather, it places the mesh object(s) in the scene and identifies the streaming data associated with these objects. Geometry and appearance of a mesh object are indicated using an IndexedFaceSet2D node and e.g. the ImageTexture node (other nodes may be used for the texture, such as MovieTexture), as is specified in ISO/IEC 14496-1. Decoding of the coded image data is performed by a Video Object decoder (this could also be a Still Texture Object decoder); decoding of coded dynamic mesh data is performed by a Mesh Object decoder, as is specified in ISO/IEC 14496-2. The decoded mesh data is used to update the appropriate fields of an IndexedFaceSet2D node. Then, the compositor uses the data of the IndexedFaceSet2D node and the decoded image data to render a texture-mapped image at specific intervals. Composition is specified in ISO/IEC 14496-1.

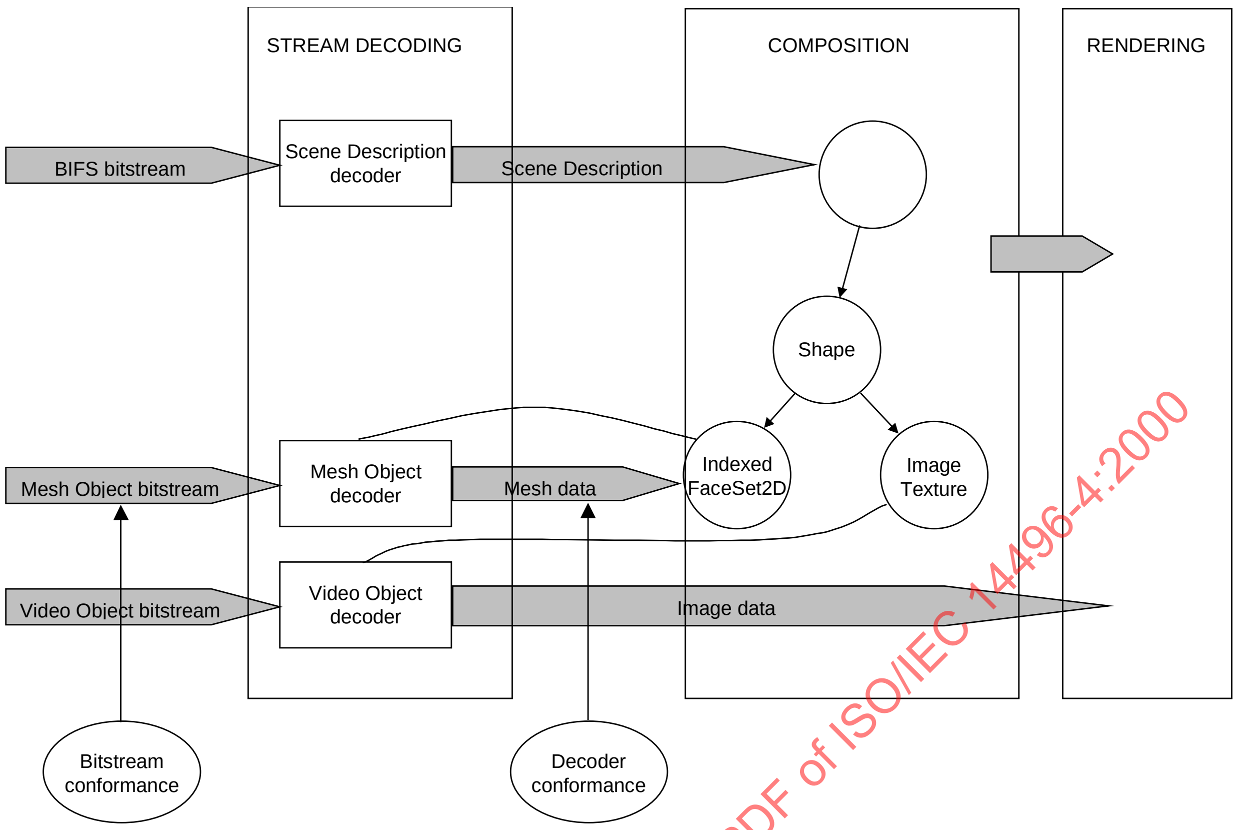

Figure 7-1 — Model for decoding and composition of mesh-based texture-mapped visual objects. Conformance points are indicated.

The conformance points for an animated 2D mesh object are, as illustrated:

1. Mesh Object bitstream conformance,

2. Mesh Object decoder conformance.

Bitstream syntax and semantics and bitstream decoding for Video Objects, Texture Objects and Mesh Objects are specified in ISO/IEC 14496-2. The structure of the coded mesh data is described in subclause 6.1.4. A Mesh Object consists of Mesh Object Planes (MOPs); a MOP corresponds to a single decoded mesh at a particular time instance. Mesh Object Planes can either be of the intra-type (I-MOP) or prediction-type (P-MOP). Initially, the mesh decoder decodes compressed binary mesh data, describing the mesh geometry (an I-MOP). Subsequently, the mesh decoder decodes compressed binary mesh data describing mesh node point displacements and computes subsequent meshes (P-MOPs). Mesh Object bitstream syntax is specified in ISO/IEC 14496-2, subclause 6.2.10. Mesh Object semantics is specified ISO/IEC 14496-2, subclause 6.3.10. Mesh object decoding is specified in ISO/IEC 14496-2, subclause 7.10.

We refer to clause 4 regarding conformance points for Video Objects and Still Texture Objects. We refer to clause 3 regarding Systems conformance points. Further composition issues with regard to 2D animated mesh objects will be discussed below.

In the following, the term bitstream refers to a Mesh Object bitstream as specified in ISO/IEC 14496-2. Also, the term decoder refers to an embodiment of the Mesh Object decoding process as specified in ISO/IEC 14496-2.

7.2.2.2.2 Related Systems BIFS Nodes, Elementary Streams

Geometry of a mesh object is indicated in BIFS by the use of an IndexedFaceSet2D node. The object appearance may further be determined by the presence of an ImageTexture or MovieTexture node.

It should be noted that compressed mesh object data may be part of a BIFS animation stream, as is specified in ISO/IEC 14496-1. In that case, an AnimationStream node shall be part of the scene description additionally.

7.2.2.2.3 Profile and Level Definitions

Visual Object Types and Visual Profiles related to 2D Mesh Animation are defined in ISO/IEC 14496-2 subclause 9.1 defines the following Visual object types that involve Mesh Object data:

1. Basic Animated Texture object type,

2. Animated 2D Mesh object type.

The first object type only allows Mesh Objects with uniform topology; the second object type allows Mesh Objects with both uniform and object-based (Delaunay) topology.

Visual profiles that include the above visual object types are defined in ISO/IEC 14496-2, subclause 9.2. These profiles are:

1. Basic Animated Texture profile,

2. Hybrid profile.

The Basic Animated Texture profile only allows Basic Animated Texture object types; the Hybrid profile allows both Basic Animated Texture object types and Animated 2D Mesh object types. 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”.

7.3 Testing Conditions

7.3.1 Description of Test Data

7.3.1.1 CD-ROM & Textual Test Data

SNHC conformance test data is supplied in the MP4 format on the attached CD-ROM.

7.3.1.2 Face Animation

In the facial animation subclause 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.

7.3.1.2.1 Definition of facial animation 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. 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.

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 shall be decodable by any compliant ISO/IEC 14496 facial animation decoder that supports the profile-and-level combination specified in the bitstream.

7.3.1.2.2 Procedure for testing bitstream conformance

ISO/IEC 14496-5 contains the source code of two software facial animation verifiers that check that a bitstream implements properly most of the normative requirements defined in ISO/IEC 14496-2. A bitstream that claims conformance with this standard shall pass the following normative test:

When processed by the technical report verifier, the bitstream shall not cause any error or non-conformance messages to be reported by the verifier. This test shall be applied only to bitstreams that are known to be free of errors introduced by transmission. Successfully passing the technical report verifier 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. 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 facial animation bitstream verifiers that perform more complete tests than those implemented by the technical report software. ISO/IEC 14496-2 contains several informative recommendations. When testing a bitstream for conformance, 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.

7.3.1.2.3 Definition of facial animation decoder conformance

In this subclause, except where stated otherwise, the term *'bitstream'* means compliant ISO/IEC 14496 facial animation bitstream (as defined in this part of ISO/IEC 14496) that has the Profile@Level indication corresponding to the Profile@Level combination considered for the decoder.

Conformance of a decoder is defined only with regard to a legal Profile@Level combination specified in clause 9 of ISO/IEC 14496-2. The normative tests that a decoder shall pass in order to claim conformance with a given profile-and-level combination are specified in subclause 7.3.1.2.4. A decoder can claim conformance 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@ML decoder).

A decoder that fails the normative tests defined by this specification may only claim limited accuracy conformance to the standard. A limited accuracy decoder is not a compliant decoder and may only be called "ISO/IEC 14496-2 limited accuracy *Profile@Level* decoder" or simply "limited accuracy *Profile@Level* decoder". In the following text, decoder conformance 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 combination are defined in clause 9 of ISO/IEC 14496-2.

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 subclause 7.3.1.2.4. 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 technical report software verifier (ISO/IEC 14496-5). The reference decoder is a decoder that implements precisely the decoding process as specified in ISO/IEC 14496-2.

7.3.1.2.4 Procedure to test decoder conformance

In this subclause, except where stated otherwise, the term *'bitstream'* means compliant ISO/IEC 14496 facial animation bitstream (as defined in this document), that has the Profile@Level indication corresponding to the Profile@Level combination for which conformance of the decoder is considered. A text version of the facial animation sequences with FAP used to generate the bitstreams is supplied in Annex [TBD] of this part of ISO/IEC 14496. MP4 test files are provided for decoder conformance testing. The decoded bitstream results have to be the same as obtained by one of the software verifiers specified in ISO/IEC 14496-5. The correct decoded FAP values are provided for each test bitstream in a simple ASCII file format with suffix "fap". If the terminal includes BIFS Face nodes, the FAP values can be observed within the BIFS scene dump file for each frame. If left-right interpolation is used or the FIT, Viseme, or Expression nodes are implemented in the terminal, the FAP values which are still set for interpolation in the FAP node after FAP decoding may subsequently be updated. Decoded FAP files are provided for both decoder output FAP values (suffix dec.fap) and final FAP node values(suffix node.fap). If only decoder output FAP values are provided for a given test, the final FAP node values are assumed to be the same. The final FAP node values provided assume that no FIT is used. The following tests are used to test decoder conformance:

7.3.1.2.4.1 Test #1

Purpose: In the frame based mode, exercise the use of the minimum and maximum values for each low level FAP (3-68) in I frames and the maximum difference a FAP can have to its value in the previous frame. Furthermore, all quantizer step sizes are executed.

File (bitstream): MinMaxFAPs.mp4

Frame-rate: 1 Hz

FAPs used: all low-level FAPs

Decoded FAP file: MinMaxFAPs.dec.fap

7.3.1.2.4.2 Test #2

Purpose: Exersize the use of start codes in I and P frames.

File (bitstream): FAPStartCode.mp4

Frame-rate: 15

FAPs used: Marco15.fap

Decoded FAP file: FAPStartCode.dec.fap

7.3.1.2.4.3 Test #3

Purpose: Decoding of various MPEG-4 FAP bitstreams

File(s) (bitstreams): Wow25.mp4, Marco30.mp4, opvis3.mp4, Expressions.mp4, Emotions.mp4, Opossum.mp4, Allfaps.mp4

FAPs used: NA

Decoded FAP file: Wow25.dec.fap, Marco30.dec.fap, opvis3.dec.fap, Expressions.dec.fap, Emotion.dec.fap, Opossum.dec.fap, Allfap.dec.fap

BIFS FAP node file: Wow25.node.fap, Marco30.node.fap, opvis3.node.fap, Expressions.node.fap, Emotion.node.fap, Opossum.node.fap, Allfap.node.fap

7.3.1.2.4.4 Test #4

Purpose: Test the ability of the FAP decoder to process bitstreams with bitrates which are just under the maximum bitrate for each level (16kbps for level 1 and 32 kbps for level 2).

File (bitstream): FAPMaxBitrate16.mp4, FAPMaxBitrate32.mp4

Frame-rate: 72 Hz FAP decoder rate with 15 Hz (for level 1) and 30 Hz (for level 2) face render frame rate

FAPs used: TBD

Decoded FAP file: FAPMaxBitrate16.dec.fap, FAPMaxBitrate32.dec.fap

7.3.1.2.4.5 Test #5

Purpose: Test Left-right and right-left interpolation of FAPs after decoding only one side (left or right) of FAPs.

File (bitstream): AllFaps_rlint.mp4

Frame-rate: 15 Hz

FAPs used: all

Decoded FAP file: AllFaps.rlint.dec.fap

BIFS FAP node file: AllFaps.rlint.node.fap

7.3.1.2.4.6 Test #6

Purpose: Test the decoding of high level FAPs 1 and 2(visemes and expressions). If the Viseme or Expression BIFS nodes are required by other MPEG-4 profiles, the complete implementation and interpretation of FAPs 1 and 2 is required in the terminal. If the Viseme and Expression BIFS nodes are not required in the terminal, the decoded values associated with FAPs 1 and 2 may be ignored. In all cases, the decoder must output the correct values.

File (bitstream): VisExp.mp4

Frame-rate: 72Hz FAP decode rate with 15 Hz (for level 1) and 30 Hz (for level 2) face render frame

FAPs used: FAP 1 & 2

Decoded FAP file: VisExp.dec.fap

BIFS FAP node file: VisExp.node.fap

7.3.1.2.4.7 Test #7

Purpose: Test the default min/max range values of the arithmetic decoder when no min/max values are included in the bitstream.

File (bitstream): NoFAPMinMax.mp4

Frame-rate: 15Hz

FAPs used: all

Decoded FAP file: NoFAPMinMax.dec.fap

7.3.1.2.4.8 Test #8

Purpose: Test the use of the DCT coding mode with I and P frames

File (bitstream): FAPDCT.mp4

Frame-rate: 15Hz

FAPs used: all FAPs

Decoded FAP file: FAPDCT.dec.fap

1.1.1.1.1.1 Test #9

Purpose: Test the use of frame-based and DCT-based coding modes in the same sequence

File (bitstream): FrameDCT.mp4

Frame-rate: 15Hz

FAPs used: all FAPs

Decoded FAP file: FrameDCT.dec.fap

1.1.1.1.1.2 Test #10 (level 2 only)

Purpose: Test multiple faces

File (bitstream): FourFaces.mp4

Frame-rate: 72 Hz FAP decode framerate and 15 Hz face rendering rate for each face

FAPs used: NA

1.1.1.1.1.3 Test #11

Purpose: Test FIT node interpolation if BIFS is included in the terminal

File (bitstream): FIT.mp4

Frame-rate: 15 Hz

FAPs used: all

Decoded FAP file: FIT.dec.fap

BIFS FAP node file: FIT.node.fap

7.3.1.2.5 Definition of animation requirement

In order for a system to be an MPEG-4 compliant facial animation system, the rendered MPEG-4 compliant face model must show all FDP feature points unless the face occludes a given point. In addition, for performing the test to verify the animation requirement the minimum size of the reference dimensions of the neutral face is specified in Table 7-2 in screen pixels.

Table 7-2 — Minimum size of the reference dimensions used to compute the FAPU

Face Model Dimensions		Minimum Size in Screen Pixels
IRISD0 = 3.1.y – 3.3.y = 3.2.y – 3.4.y	Iris diameter (by definition equal to the distance between upper and lower eyelid) in neutral face	4
ES0 = 3.5.x – 3.6.x	Eye separation	30
ENS0 = 3.5.y – 9.15.y	Eye - nose separation	40
MNS0 = 9.15.y – 2.2.y	Mouth - nose separation	20
MW0 = 8.3.x – 8.4.x	Mouth width	25

The FAP bitstream *AllFaps* is used to perform the test. The facial Feature Points are required to move with respect to the FAPs in the bitstream as defined in ISO/IEC 14496-2.

In order to verify if the FAPs are correctly interpreted, it is advised to use one of the facial animation systems software that are part of ISO/IEC 14496-5 and make a subjective test by a side to side comparison between the facial animation system subjected to the test and one of these facial animation systems.

The 3D co-ordinates of the decoder’s generic facial model are expressed in meters and it is recommended that the size should correspond to a natural size of a human head, insuring a reasonable composition of the face with a 3D scene.

In case where the face is used outside of the context of a 3D Graphics Profile, it is recommended that the rendered face should occupy 80% of the screen height and be located in the centre of the screen when no transformation is applied to it. This insures that the face can be reasonably placed in 2D scenes.

7.3.1.3 2D Mesh Animation

7.3.1.3.1 Definition of Mesh Object bitstream conformance

A conforming bitstream implements all normative parts of ISO/IEC 14496-2 and meets all syntactic and semantic requirements of that specification, including any profile-and-level restrictions defined in that specification that invoke 2D Mesh Animation functionality.

Specific or additional requirements on conforming bitstreams are the following.

The field **mesh_type_code** shall be set to '01' in bitstreams conforming to the Basic Animated Texture profile. This field shall be set to either '01' or '10' in bitstreams conforming to the Hybrid profile.

The value of **nr_of_mesh_nodes_hor** must be >= 2 (if **mesh_type_code** == 01).

The value of **nr_of_mesh_nodes_vert** must be >= 2 (if **mesh_type_code** == 01).

The value of **mesh_rect_size_hor** must be > 0 (if **mesh_type_code** == 01).

The value of **mesh_rect_size_vert** must be > 0 (if **mesh_type_code** == 01).

The value of **nr_of_mesh_nodes** must be >= 3 (if **mesh_type_code** == 10).

The value of **nr_of_boundary_nodes** must be >= 3 (if **mesh_type_code** == 10).

The value of (**frame_rate** + **seconds**/16) must be > 0 (if **is_intra** == 1 and **is_frame_rate** == 1).

The bitstream shall not contain any sequence of 6 or more consecutive Mesh Object Planes with **is_intra** = 0 and **mesh_mask** = 0 and **skip_frames** = 0 and without a **mesh_object_plane_start_code**.

7.3.1.3.1.1 Additional encoder requirements

Although encoders are not directly addressed by ISO/IEC 14496-2, an encoder is said to be a conforming ISO/IEC 14496-2 Mesh Object encoder if it satisfies the following requirements:

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

2. In case the coded mesh has Delaunay topology, the Delaunay triangulation shall conform to the description given in subclause 7.11.1.2 of ISO/IEC 14496-2.

This second requirement is necessary to guarantee that compliant decoders will produce Mesh Objects with correct topology and correct geometry over all I and P Mesh Object Planes.

7.3.1.3.2 Procedure for testing Mesh Object bitstream conformance

ISO/IEC 14496-5 contains the source code of a reference software Mesh Object decoder that checks that a bitstream implements properly most of the normative requirements defined in ISO/IEC 14496-2.

A bitstream that claims conformance with this standard shall pass the following normative test:

When processed by the reference decoder of ISO/IEC 14496-5, the bitstream shall not cause any error or non-conformance messages to be reported by the software. This test shall be applied only to bitstreams that are known to be free of errors introduced by transmission.

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 Mesh Object bitstream decoders or verifiers that perform more complete tests than those implemented by the reference software of ISO/IEC 14496-5.

7.3.1.3.3 Definition of Mesh Object decoder conformance

Here, the term decoder refers to an embodiment of the Mesh Object decoding process as specified in ISO/IEC 14496-2. A conforming Mesh Object decoder is a decoder that passes the normative tests defined in subclause 7.3.1.4. Decoder conformance is defined in two parts:

1. Non-real-time (functional) conformance,

2. Real-time (performance) conformance.

Non-real-time conformance demands that a decoder shall correctly decode a conforming Profile@Level Mesh Object bitstream. Real-time conformance in addition demands that a decoder shall output decoded Mesh Object planes to the composition process at the rate or within the time period indicated in the bitstream.

7.3.1.3.4 Procedure for testing Mesh Object decoder conformance

Decoder conformance is determined using test suites, consisting of test bitstreams, (a) reference decoder(s) and associated measurement algorithms. Generally, the decoder to be tested and the reference decoder are both supplied with a test bitstream and the output of the decoder to be tested shall be compared with the output of the reference decoder using the measurement algorithm(s). The measurement algorithms apply the set of conformance tests as defined in subclause 7.3.1.4. A measurement algorithm verifies, for instance, that the locations of mesh node points decoded by the tested decoder coincide with the locations of corresponding mesh node points decoded by the reference decoder. The test bitstreams, listed in subclause 7.3.1.3.5, shall stress the decoder on both functionality and performance.

7.3.1.3.5 Test bitstream summary

A test suite is provided that includes the bitstreams for animation of 2D uniform meshes and object-based (Delaunay) meshes. Test bitstreams generally are used to test either non-real-time (functional) conformance or real-time (performance) conformance. Particular test bitstreams are designed to exercise particular functionalities or complexities, which are listed in the following subclause.

7.3.1.3.6 Test bitstream functionality and complexity

Bitstreams to test non-real-time conformance generally test the ability of the decoder to parse the bitstream syntactically correct and interpret its elements semantically correct. More specifically, the following shall be tested, where associated elements of the Mesh Object syntax are indicated in square brackets:

a. Semi-random insertion of optional start-codes [mesh_object_plane_start_code] in series of I and P Mesh Object Planes;

b. Exploring all options and parameter ranges of the temporal header part of the Mesh Object Plane header, such as:

   - Time codes [time_code];
   - Frame rates [frame_rate, seconds, frequency_offset] ;
   - Frame skipping [number_of_frames_to_skip];

c. Both options to code mesh geometry [mesh_type_code], i.e., meshes with

   - Uniform topology/geometry;
   - Object-based (Delaunay) topology/geometry;

d. Exploring the ranges of the mesh geometry parameters in I-Mesh Object Planes, specifically:

   - Number of node points (vertices) [nr_of_mesh_nodes_hor, nr_of_mesh_nodes_vert, nr_of_mesh_nodes, nr_of_boundary_nodes];
   - VLC bounds (sizes of node point coordinate differences) [delta_x_len_vlc, delta_x, delta_y_len_vlc, delta_y];

e. Correctness of mesh geometry decoding and topology reconstruction, specifically in the case of object-based (Delaunay) meshes;

f. Exploring the ranges of the mesh motion parameters in P-Mesh Object Planes, specifically:

   - Motion vector range [motion_range_code];
   - VLC bounds (sizes of node point motion vector differences) [delta_mv_x_vlc, delta_mv_x_res, delta_mv_y_vlc, delta_mv_y_res];

g. Correctness of mesh motion decoding, specifically as it relates to mesh triangle traversal.

Bitstreams to test real-time conformance generally are bitstreams wherein each or all of the following parameters are maximized, while conforming to the restrictions specified by a profile and level:

a. Number of Mesh objects;

b. Mesh object frame rate;

c. Number of node points (vertices) of Mesh object planes;

d. Range of node point motion vectors.

These are also the main parameters that determine the Mesh object bitstream bit rate.

7.3.1.3.7 Test bitstream specification

The following table specifies the bitstreams that shall be used to test non-real-time (functional) conformance; where the specified functionalities refer to those listed in subclause 7.3.1.3.6.

Table 7-3 — Bitstreams used to test functionality

Test bitstream	Topology type	Functionalities tested	Profiles	Number of MOPs
mesh01	uniform	a	Basic Animated Texture & Hybrid	120
mesh02	uniform	c; b	Basic Animated Texture & Hybrid	155
mesh03	object-based	c; b	Hybrid	155
mesh04	uniform	c; d; f	Basic Animated Texture & Hybrid	136
mesh05	object-based	c; d; f	Hybrid	133
mesh06	uniform	c; e; g	Basic Animated Texture & Hybrid	100
mesh07	object-based	c; e; g	Hybrid	100

Note that all bitstreams test, to some extent, correctness of mesh geometry and motion decoding, but bitstreams ’mesh06’ and ’mesh07’ do so to a fuller extent.

The following table specifies the bitstreams that shall be used to test real-time performance. Each bitstream tests conformance to a certain profile and level. Each bitstream contains 300 Mesh Object Planes.

Table 7-4 — Bitstreams used to test real-time performance

Test bitstream	Mesh topology type	Number of node points	Frame rate	Bit rate	Profile and level
mesh08	uniform	120	30 Hz	16 kbit/s	Basic Animated Texture & Hybrid @ level 1
mesh09	uniform	480	30 Hz	64 kbit/s	Basic Animated Texture & Hybrid @ level 1
mesh10	uniform	437	60 Hz	64 kbit/s	Basic Animated Texture & Hybrid @ level 2
mesh11	uniform	1748	60 Hz	128 kbit/s	Basic Animated Texture & Hybrid @ level 2
mesh12	object-based	120	30 Hz	16 kbit/s	Hybrid @ level 1
mesh13	object-based	480	30 Hz	64 kbit/s	Hybrid @ level 1
mesh14	object-based	437	60 Hz	64 kbit/s	Hybrid @ level 2
mesh15	object-based	1748	60 Hz	128 kbit/s	Hybrid @ level 2

7.3.1.4 Performance and functionality conformance tests

Functionality (non-real-time) conformance shall be tested using (a) measurement algorithm(s), which generally compare(s) the output of the reference decoder with that of the decoder to be tested. The output of both decoders shall contain the following data elements for each Mesh object plane:

- Number of mesh node points;
- Number of mesh triangles;
- For each mesh node point, the coordinate pair (x, y);
- For each mesh triangle, the triplet of node point indices (i, j, k).

For, performance (real-time) conformance testing, the output of the decoders in addition shall contain for each Mesh object plane:

- Timing of the output.

The tests that a conforming non-real-time decoder shall pass in order to be able to claim conformance are the following:

1. Each Mesh object plane is output in the correct order with respect to other (previous and subsequent) Mesh object planes.

2. The number of mesh node points of the output Mesh object plane of the reference and test decoder must be equal;

3. The number of mesh triangles of the output Mesh object plane of the reference and test decoder must be equal;

4. The node points of the output Mesh object plane of the test decoder shall be unique, i.e., there shall be no two node points with the same (*x*, *y*) coordinates. Each node point can be assigned a unique index, but note that there is no restriction on the index of ordering of the node points;

5. For every node point of the output Mesh object plane of the test decoder, there shall be a node point of the output Mesh object plane of the reference decoder with the same (*x*, *y*) coordinates;

6. For every triangle of the output Mesh object plane of the test decoder, there shall be a triangle of the output Mesh object plane of the reference decoder that refers to three node points with the same (*x*, *y*) coordinates (note that the indices of the node points do not have to correspond).

The tests that a conforming real-time decoder shall pass in order to claim conformance are, in addition to the above five tests, the following:

7. Mesh object planes are decoded from the bitstream within the time period indicated in the bitstream, by time codes and/or by a frame rate. That is, all the required data elements shall be available at the output of the decoding process within the time period indicated.

Annex A
(informative)

Sample Bank Format (SASBF) compliance testing and materials

A.1 Introduction

This Annex documents the test sets generated for ISO/IEC 14496-3 subpart 5 Structured Audio Sample Bank Format (SASBF) compliance testing. Three components form a single test set: SASBF bank, MIDI sequence and the reference output waveform generated by the reference synthesizer (.wav file). SASBF files were generated using the software AWAVE [References: Awave], MIDI sequences are MIDI format 0 (.mid files) generated by a MIDI sequencer [References: MIDI]. These two components form the operand of the non-real time reference synthesizer program, a DOS executable file that generates the output .wav file.

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.

Each test will be 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.

A.2 Organization

Figure A-1 presents the abstraction of test procedures, organized into three categories as depicted. The work described herein has been confined to the DLS2 Default Connections, which form the largest portion of the test set. Figure A-2 illustrates the approach that was taken in creating test routines for default connections, which resulted in the organization of test banks and directory shown in Table A-1. The Limits folder consist of test for the specified maximum and minimum values of the Low Frequency Oscillator (LFO) and Envelope Generator (EG). Tests for filters were designed to demonstrate behavior at the 22.05kHz, 44.1kHz and 48kHz sampling rates.

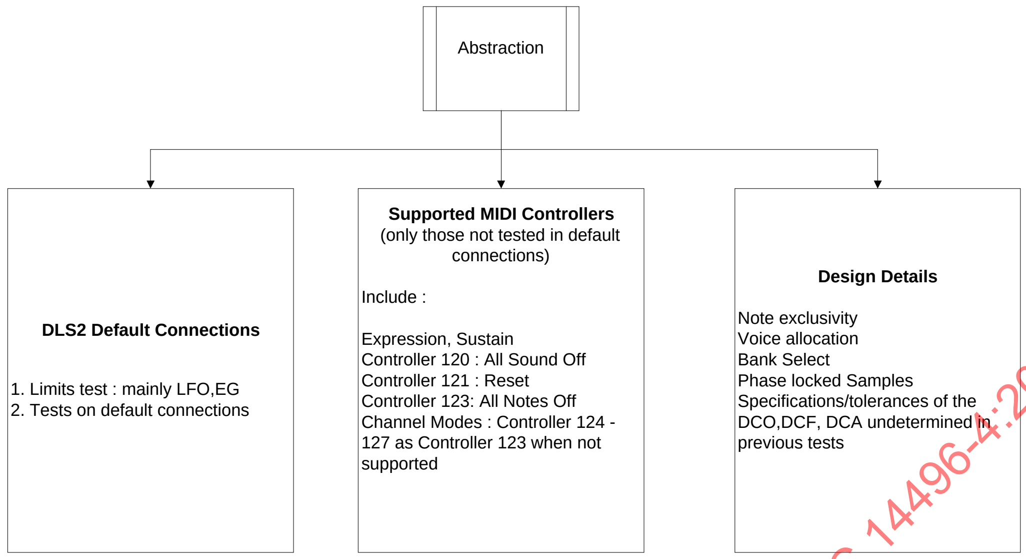
Abstraction
DLS2 Default Connections
1. Limits test : mainly LFO,EG
2. Tests on default connections
Supported MIDI Controllers
(only those not tested in default connections)
Include :
Expression, Sustain
Controller 120 : All Sound Off
Controller 121 : Reset
Controller 123: All Notes Off
Channel Modes : Controller 124 - 127 as Controller 123 when not supported
Design Details
Note exclusivity
Voice allocation
Bank Select
Phase locked Samples
Specifications/tolerances of the DCO,DCF, DCA undetermined in previous tests

Figure A-1 — Abstraction of Test Procedures

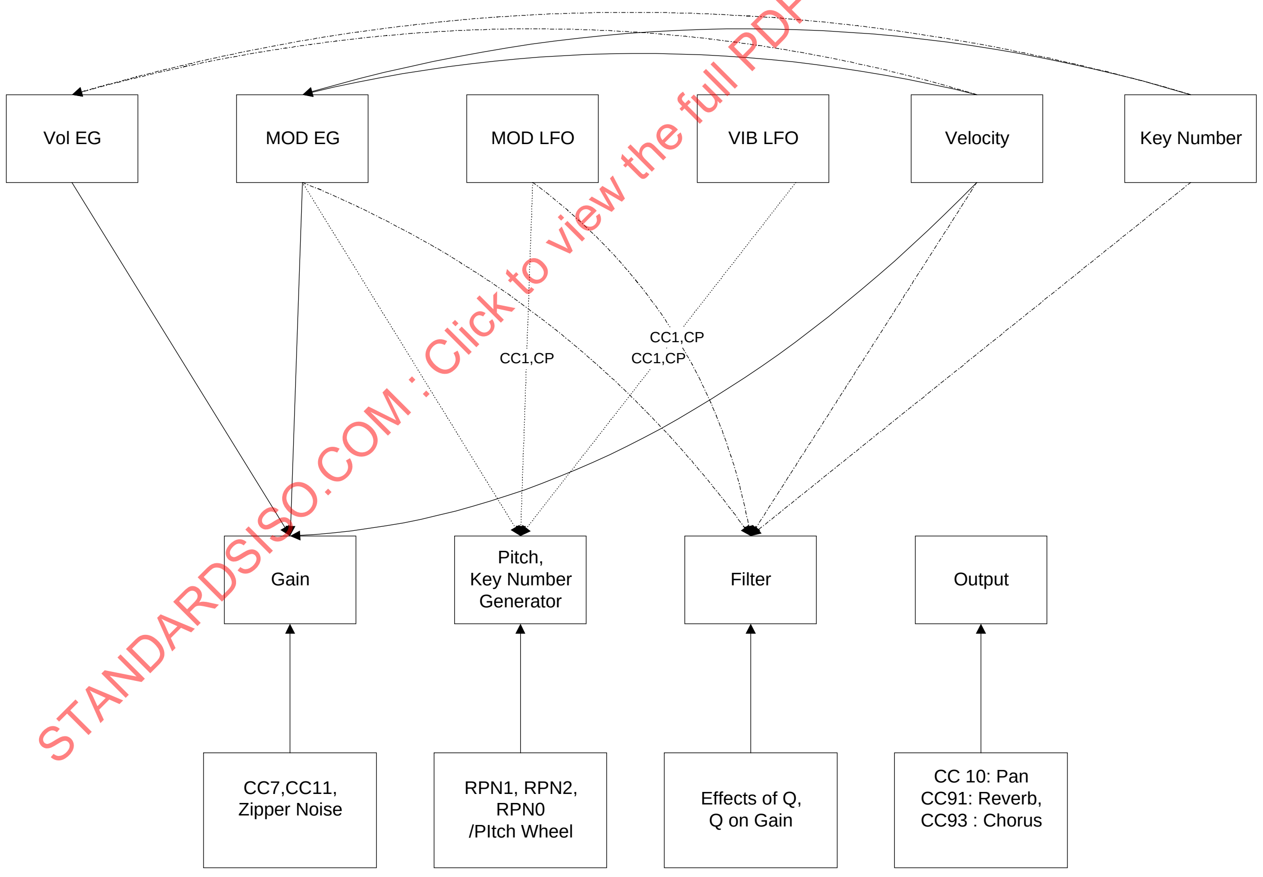

Figure A-2 — Default Connections

Table A-1 — Test Banks

Bank No	Description
1	Limits\lfo\LimitsLFO.dls
2	Limits\VolEG\LimitsVolEG.dls
3	Limits\ModEG\LimitsModEG.dls
4	Pitch\PitchEG\PitchEG.dls
5	Pitch\PitchModLFO\PitchModLFO.dls
6	Pitch\VibModLFO\VibModLFO.dls
7	Pitch\PitchGen\PitchGen.dls
8	Gain\GainModLFO\GainModLFO.dls
9	Gain\GainEG\GainEG.dls
10	Gain\GainGen\GainGen.dls
11	Filter22\FilterGen\FilterGen22.dls
11	Filter44\FilterGen\FilterGen44.dls
11	Filter48\FilterGen\FilterGen48.dls
12	Filter22\FilterModLFO\FilterModLFO22.dls
12	Filter44\FilterModLFO\FilterModLFO44.dls
12	Filter48\FilterModLFO\FilterModLFO48.dls
13	Filter22\FilterEG\FilterEG22.dls
13	Filter44\FilterEG\FilterEG44.dls
13	Filter48\FilterEG\FilterEG48.dls

A.3 Test Sequence Structure

The design of MIDI files for test sequences can be based either on absolute time or beats. **A fixed tempo is required for the former, which is 100bpm (beats per minute) for the generated test sequences**. Design based on absolute time was required when time measurement is essential, for example, in the case of measuring the attack time of an envelope. In most sequences, there will be a 500ms interval between two note-on events. Exclusions will be documented with EIO (Events Interval Omitted). With particular reference to test sequences for EG, a test may be designed to include a '*Post Listening*' period for easy measurement. Consider if the attack time of an envelope is to be measured, the sustain segment could be set to zero such that the end of the attack segment could be easily determined. This '*Post Listening*' period was abbreviated PLT in the documentation and is 500ms unless specified otherwise. Unless specified, test sequences were also designed in the order according to the each bank's patch number.

A.4 Abbreviations

In view of a concise tabulation of banks patches, abbreviations were used frequently. Table A-2 provides a list of abbreviations used in this report.