
**Information technology — Coding of
audio-visual objects —**

**Part 4:
Conformance testing**

*Technologies de l'information — Codage des objets audiovisuels —
Partie 4: Essai de conformité*



PDF disclaimer

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

Adobe is a trademark of Adobe Systems Incorporated.

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

© ISO/IEC 2004

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either ISO at the address below or ISO's member body in the country of the requester.

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

Published in Switzerland

Contents

	Page
Foreword	vii
Introduction	viii
1 Scope	1
2 Normative references	1
3 Terms and definitions	2
4 Systems	3
4.1 Conformance Points	3
4.1.1 FlexMux Conformance Point	4
4.1.2 Sync Layer Conformance Point	4
4.1.3 OD Conformance Point	4
4.1.4 BIFS Conformance Point	4
4.1.5 OCI Conformance Point	4
4.1.6 IPMP Conformance Point	4
4.1.7 Scene Graph Conformance Point	4
4.2 Bitstream Conformance	4
4.2.1 FlexMux Conformance	5
4.2.2 Synchronization Layer Conformance	5
4.2.3 OD Conformance	5
4.2.4 BIFS Conformance	5
4.2.5 OCI Conformance	5
4.2.6 IPMP Conformance	6
4.2.7 Miscellaneous Conformance	6
4.3 Terminal Conformance	6
4.3.1 FlexMux conformance	7
4.3.2 Synchronization Layer Conformance	7
4.3.3 OD Conformance	10
4.3.4 BIFS Conformance	13
4.3.5 OCI Conformance	14
4.3.6 IPMP Conformance	14
4.3.7 Scene Graph Conformance	14
4.3.8 Miscellaneous Conformance	15
4.4 Test material and test suites	15
4.4.1 Parsing Hint File Format	16
4.4.2 Scene Dump File Format	18
4.4.3 Test Suites	20

4.5	Advanced BIFS	27
4.5.1	Bitstream conformance	27
4.5.2	Terminal conformance	27
4.6	MPEG-J	28
4.6.1	MPEG-J Conformance Points	28
4.6.2	Bitstream Conformance	29
4.6.3	Terminal Conformance	29
4.7	MP4 File Format	30
4.7.1	Writing	30
4.7.2	Reading	31
5	Visual	31
5.1	Introduction	31
5.2	Definition of visual bitstream compliance	32
5.2.1	Requirements and restrictions related to profile-and-level	32
5.2.2	Additional restrictions on bitstream applied by the encoder	32
5.2.3	Encoder requirements and recommendations	32
5.3	Procedure for testing bitstream compliance	33
5.4	Definition of visual decoder compliance	34
5.4.1	Requirement on arithmetic accuracy in video objects (without IDCT)	34
5.4.2	Requirement on arithmetic accuracy in video objects (with IDCT)	35
5.4.3	Requirement on arithmetic accuracy in scalable still texture object (without IDWT)	35
5.4.4	Requirement on arithmetic accuracy in scalable still texture (with IDWT)	36
5.4.5	Requirement on output of the decoding process and timing	36
5.4.6	Recommendations	36
5.5	Procedure to test decoder compliance	36
5.5.1	Static tests	36
5.5.2	Dynamic tests	37
5.5.3	Specification of the test bitstreams	37
5.5.4	Implementation of the static test	51
5.5.5	Implementation of the dynamic test	52
5.5.6	Decoder conformance	52
5.5.7	Normative Test Suites for Simple, Simple Scalable, Core, Main and N-Bit profile	52
5.5.8	Bitstream Donated by MPEG-4 Platform Verification Bitstream Development Project	55
5.6	Additional Conformance Testing	63
5.6.1	Specification of the test bitstreams	63
5.6.2	Normative Test Suites for Advanced Real-Time Simple (ARTS), Core Scalable, Advanced Coding Efficiency (ACE), Advanced Core (AC) and Advanced Scalable Texture profiles	78
6	Audio	84
6.1	Terms and Definitions	84
6.2	Introduction	84

6.3	Audio Conformance Points.....	85
6.4	Audio Profiles	86
6.5	Conformance data.....	86
6.5.1	File name conventions	86
6.5.2	Content.....	88
6.6	Audio Object Types	88
6.6.1	General	88
6.6.2	Null.....	94
6.6.3	AAC-based scalable configurations	94
6.6.4	AAC (main, LC, ER LC, SSR, LTP, ER LTP, ER LD, scalable, ER scalable)	95
6.6.5	TwinVQ and ER_TwinVQ.....	112
6.6.6	ER BSAC	115
6.6.7	CELP	119
6.6.8	ER CELP	123
6.6.9	HVXC	127
6.6.10	ER HVXC	137
6.6.11	ER HILN and ER Parametric.....	139
6.6.12	TTSI.....	153
6.6.13	General MIDI	155
6.6.14	Wavetable Synthesis	155
6.6.15	Algorithmic Synthesis and AudioFX.....	156
6.6.16	Main Synthetic.....	162
6.7	Audio EP tool.....	163
6.7.1	Compressed data	163
6.7.2	Decoders	165
6.8	Audio Composition	170
6.8.1	Introduction	170
6.8.2	Common Audio Composition Characteristic	172
6.8.3	AudioSource and Sound2D	173
6.8.4	AudioSource and Sound	175
6.8.5	AudioSwitch	175
6.8.6	AudioMix and Sampling Rate Conversion.....	176
6.8.7	AudioFX	177
6.9	MPEG-4 audio transport stream	177
6.9.1	Compressed Data.....	178
6.9.2	Decoders.....	178
6.10	Upstream.....	179
6.10.1	Compressed data	179
6.10.2	Decoders.....	179
6.11	Advanced Audio BIFS nodes.....	179

6.11.1	Introduction.....	179
6.11.2	Composition Unit Inputs.....	180
6.11.3	Compositor Output.....	180
6.11.4	Physical Approach	180
6.11.5	Perceptual Approach	191
6.12	Conformance test sequence assignment to profiles and levels	202
6.12.1	Audio	203
6.12.2	Systems	210
7	DMIF	213
7.1	Introduction.....	213
7.2	The PICS.....	214
7.2.1	Global statement of conformance	214
7.2.2	DMIF-Application Interface.....	214
7.3	The Conformance ATS.....	224
7.3.2	ATS for DAI in Remote Interactive Scenarios	225
7.3.3	ATS for DAI in Local Storage Scenarios	226
7.3.4	ATS for DAI in Broadcast Scenarios	231
8	SNHC	235
8.1	Introduction.....	235
8.1.1	Purpose & Scope.....	236
8.1.2	Intended Use of Decoders	236
8.1.3	What Is To Be Tested	236
8.2	Body Animation	236
8.2.1	Simple FBA Profile	236
8.2.2	FBA Conformance Points.....	237
8.2.3	FBA Testing Conditions	238
8.3	3D Mesh Coding	242
8.3.1	Conformance Points	243
8.3.2	Bitstream Conformance.....	243
8.3.3	Decoder Conformance.....	244
Annex A (informative)	Sample Bank Format (SASBF) compliance testing and materials	250
Annex B (informative)	Complexity measurement criteria and tool for level definitions of algorithmic synthesis and AudioFX Object Type	273
Annex C (Informative)	Test bitstreams for the CELP object type	292
Annex D (informative)	Patent statements	295
Annex E (informative)	Revised Text for Agreement with Sun Microsystems	297
Bibliography.....		298

Foreword

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

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

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

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

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

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

- *Part 1: Systems*
- *Part 2: Visual*
- *Part 3: Audio*
- *Part 4: Conformance testing*
- *Part 5: Reference software*
- *Part 6: Delivery Multimedia Integration Framework (DMIF)*
- *Part 7: Optimised reference software for coding of audio-visual objects*
- *Part 8: Carriage of ISO/IEC 14496 contents over IP networks*
- *Part 9: Reference hardware description*
- *Part 10: Advanced Video Coding*
- *Part 11: Scene description and application engine*
- *Part 12: ISO base media file format*
- *Part 13: Intellectual Property Management and Protection (IPMP) extensions*
- *Part 14: MP4 file format*
- *Part 15: Advanced Video Coding (AVC) file format*
- *Part 16: Animation Framework eXtension (AFX)*
- *Part 17: Streaming text format*
- *Part 18: Font compression and streaming*
- *Part 19: Synthesized texture stream*

Introduction

Parts 1, 2 and 3 of ISO/IEC 14496 specify a multiplex structure and coded representations of audio-visual information. Parts 1, 2 and 3 of ISO/IEC 14496 allow for large flexibility, achieving suitability of ISO/IEC 14496 for many different applications. The flexibility is obtained by including parameters in the bitstream that define the characteristics of coded bitstreams. Examples are the audio sampling frequency, picture size, picture shape, picture rate, bitrate parameters, synchronisation timestamps, the association of bitstreams and synthetic objects within objects, the association of objects within scenes, the protection of bitstreams, objects and scenes. Part 6 of ISO/IEC 14496 specifies a framework for uniform delivery of MPEG-4 content according to the requested associated QoS, irrespective of their location and the transport technology.

This part of ISO/IEC 14496 specifies how tests can be designed to verify whether bitstreams and decoders meet the requirements as specified in parts 1, 2, 3 and 6 of ISO/IEC 14496 and allow interoperability with remote terminals in interactive, broadcast and local (with stored contents) sessions. These tests can be used for various purposes such as:

- manufacturers of encoders, and their customers, can use the tests to verify whether the encoder produces bitstreams compliant with parts 1, 2 and 3 of ISO/IEC 14496.
- manufacturers of decoders and their customers can use the tests to verify whether the decoder meets the requirements specified in parts 1, 2 and 3 of ISO/IEC 14496 for the claimed decoder capabilities.
- manufacturers and customers of terminals supporting interactive, broadcast and local sessions over a multitude of transport protocols and networks, can use the tests to verify whether the claimed functionalities are compliant with ISO/IEC 14496-6.
- manufacturers of test equipments, and their customers can use the tests to verify compliance with parts 1, 2 and 3 of ISO/IEC 14496.

Information technology — Coding of audio-visual objects —

Part 4: Conformance testing

1 Scope

This part of ISO/IEC 14496 specifies how tests can be designed to verify whether bitstreams and decoders meet requirements specified in parts 1, 2 and 3 of ISO/IEC 14496 and for part 6 of ISO/IEC 14496 it specifies how tests can be designed for bitstream delivery over various delivery technologies in an interoperable transparent manner to parts 1, 2 and 3. In this part of ISO/IEC 14496, encoders are not addressed specifically. An encoder may be said to be an ISO/IEC 14496 encoder if it generates bitstreams compliant with the syntactic and semantic bitstream requirements specified in parts 1, 2 and 3 of ISO/IEC 14496.

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

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

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

2 Normative references

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

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

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

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

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

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

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

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

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

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

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

ISO/IEC 11172-3:1993, *Information technology — Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s — Part 3: Audio*

ISO/IEC 11172-4:1995, *Information technology — Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s — Part 4: Compliance testing*

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

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

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

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

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

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

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

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

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

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

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

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

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

3 Terms and definitions

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

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

4 Systems

4.1 Conformance Points

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

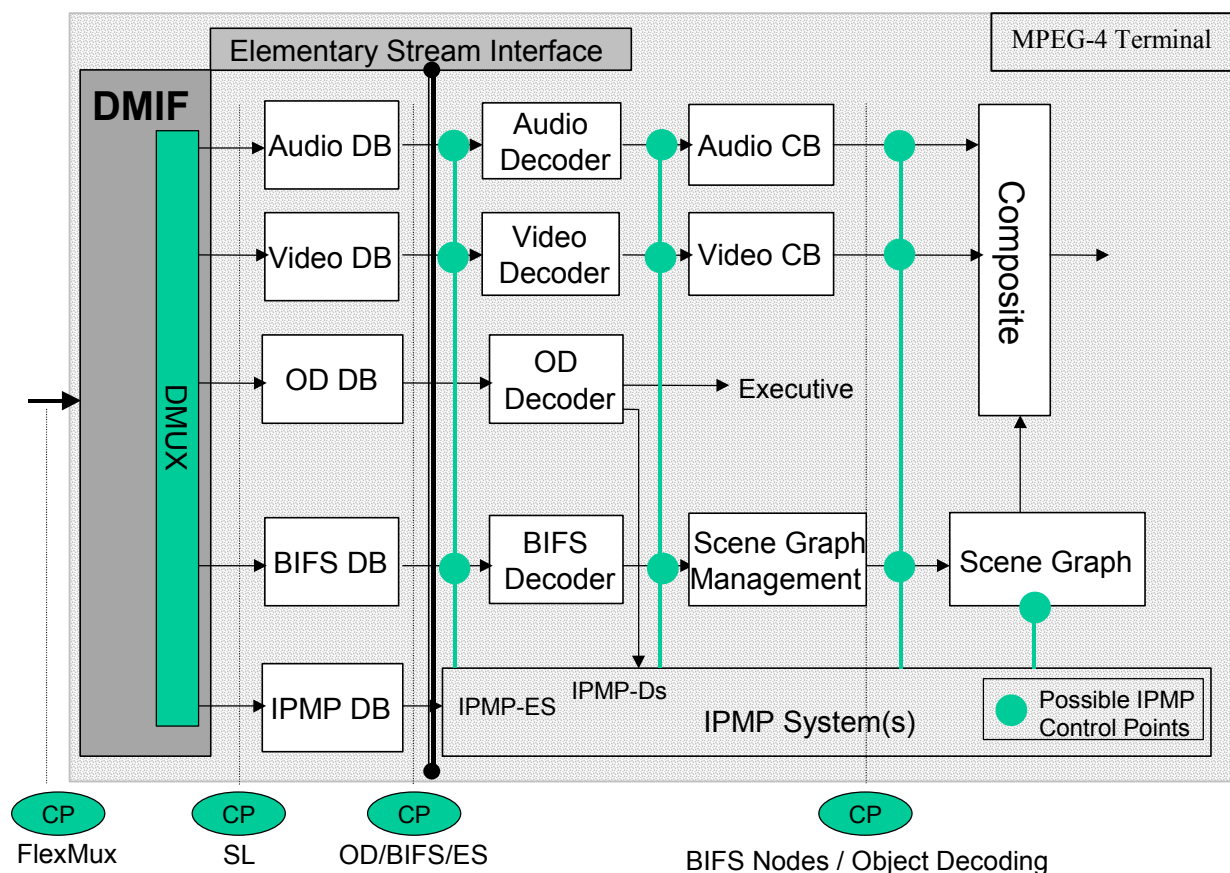


Figure 1 — Typical MPEG-4 terminal

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

Bitstream conformance points are:

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

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

Terminal conformance points are:

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

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

4.1.1 FlexMux Conformance Point

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

4.1.2 Sync Layer Conformance Point

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

4.1.3 OD Conformance Point

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

4.1.4 BIFS Conformance Point

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

4.1.5 OCI Conformance Point

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

4.1.6 IPMP Conformance Point

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

4.1.7 Scene Graph Conformance Point

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

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

4.2 Bitstream Conformance

Each bitstream shall meet the syntactic and semantic requirements specified in ISO/IEC 14496-1. This subclause describes a set of tests to be performed on bitstreams. In the description of the tests it is assumed that the tested bitstream contains no errors due to transmission or other causes. For each test the condition or conditions that must be satisfied are given, as well as the prerequisites or conditions in which the test can be

applied. Note that the application of these tests requires parsing of the bitstream to the appropriate levels. Parsing and interpretation of ODs is also required. In some cases of IPMP-protected data, de-scrambling may be required before the tests can be performed on non IPMP-related features.

4.2.1 FlexMux Conformance

4.2.1.1 Conformance Requirements

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

4.2.1.2 Measurement procedure

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

4.2.1.3 Tolerance

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

4.2.2 Synchronization Layer Conformance

4.2.2.1 Conformance Requirements

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

4.2.2.2 Measurement procedure

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

4.2.2.3 Tolerance

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

4.2.3 OD Conformance

4.2.3.1 Conformance Requirements

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

4.2.3.2 Measurement procedure

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

4.2.3.3 Tolerance

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

4.2.4 BIFS Conformance

4.2.4.1 Conformance Requirements

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

4.2.4.2 Measurement procedure

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

4.2.4.3 Tolerance

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

4.2.5 OCI Conformance

4.2.5.1 Conformance Requirements

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

4.2.5.2 Measurement procedure

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

4.2.5.3 Tolerance

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

4.2.6 IPMP Conformance

4.2.6.1 Conformance Requirements

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

4.2.6.2 Measurement procedure

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

4.2.6.3 Tolerance

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

4.2.7 Miscellaneous Conformance

4.2.7.1 Conformance Requirements

4.2.7.1.1 Private data handling

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

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

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

4.2.7.1.2 Buffer management

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

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

4.2.7.2 Measurement procedure

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

4.2.7.3 Tolerance

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

4.3 Terminal Conformance

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

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

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

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

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

4.3.1 FlexMux conformance

4.3.1.1 Conformance Requirements

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

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

4.3.1.2 Measurement procedure

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

4.3.1.3 Tolerance

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

4.3.2 Synchronization Layer Conformance

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

4.3.2.1 Conformance Requirements

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

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

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

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

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

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

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

4.3.2.1.1 The Synchronization Layer Configuration Descriptor

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

4.3.2.1.2 Structure

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

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

If predefined = 0x00:

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

4.3.2.2 Measurement Procedure

4.3.2.2.1 Timing Accuracy (OCR) Procedure

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

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

4.3.2.2.1.1 From the transmission point of view

Assumptions:

- a) the network provides a constant delay transmission for any bytes of the bitstreams.
- b) bitstreams are delivered at a constant bitrate.
- c) It is also assumed that constant bitrate means that there is a number 'r' such that for every time interval ΔT the following inequality is satisfied:

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

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

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

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

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

Table 1 — Naming Convention for OCRs

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

And as a result we have:

$$\begin{aligned}
 OCR[i] &= t_i \pm e \\
 OCR[j] &= t_j \pm \delta \\
 t_j - t_i &= [T_j - T_i] * \left[1 + \frac{\Delta f}{f} \right]
 \end{aligned}$$

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

Therefore, we can deduce:

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

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

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

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

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

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

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

and

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

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 , $\Delta f_{\max}$, f are, respectively, 500 nsec, 810 Hz, and 27MHz.

4.3.2.2.1.2 From a stored bitstream point of view

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

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

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

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

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

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

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

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:

$$const_{min} \leq const_{max}$$

the conformance test failed.

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

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

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

4.3.2.2.2 Timestamping (DTS & CTS) Procedure

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

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

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

4.3.2.3 Tolerance

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

4.3.3 OD Conformance

4.3.3.1 Conformance Requirements

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

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

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

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

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

4.3.3.1.1 Structure for URLs

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

4.3.3.1.2 The Initial Object Descriptor

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

The structural conformance is a part of the syntactic conformance.

4.3.3.1.2.1 Structure

Shall have its tag value equal to 0x02.

Shall have an ObjectDescriptorID value not equal to 0x000.

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

- ODProfileLevelIndication
- sceneProfileLevelIndication
- audioProfileLevelIndication
- visualProfileLevelIndication
- graphicsProfileLevelIndication

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

The InitialObjectDescriptor may aggregate at most 30 ES_Descriptor elements.

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

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

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

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

4.3.3.1.2.2 Scope of URLs in the Initial Object Descriptor

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

4.3.3.1.3 The Object Descriptor

4.3.3.1.3.1 Structure

An object descriptor shall be encapsulated within an ObjectDescriptorUpdate command.

Shall have the tag value equal to 0x01.

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

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

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

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

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

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

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

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

4.3.3.1.3.2 Aggregation of ES_Descriptors in an Object Descriptor

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

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

4.3.3.1.4 Scope of URLs in Object Descriptors

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

4.3.3.1.5 Elementary Stream Descriptors

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

4.3.3.1.5.1 Structure

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

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

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

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

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

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

An ES_Descriptor shall aggregate one DecoderConfigDescriptor and one SLConfigDescriptor.

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

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

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

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

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

4.3.3.1.5.2 Elementary Stream Dependencies

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

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

4.3.3.1.5.3 Scope of URLs in ES_Descriptors

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

4.3.3.1.5.4 Name Scope of Identifiers

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

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

4.3.3.1.5.5 Reuse of identifiers

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

4.3.3.1.6 Decoder Configuration Descriptors

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

4.3.3.1.6.1 Structure

The `DecoderConfigDescriptor` shall have a tag value of 0x04.

The `objectTypeIndication` value shall not be 0x00.

The `streamType` value shall not be 0x00.

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

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

4.3.3.2 Tolerance

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

4.3.4 BIFS Conformance

4.3.4.1 Conformance Requirements

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

4.3.4.2 Measurement Procedure

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

4.3.4.3 Tolerance

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

4.3.5 OCI Conformance

4.3.5.1 Conformance Requirements

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

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

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

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

4.3.5.2 Measurement procedure

This procedure is application dependant.

4.3.5.3 Tolerance

The tolerance is application dependant.

4.3.6 IPMP Conformance

4.3.6.1 Conformance Requirements

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

4.3.6.2 Measurement Procedure

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

4.3.6.3 Tolerance

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

4.3.7 Scene Graph Conformance

4.3.7.1 Conformance Requirements

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

4.3.7.2 Measurement procedure

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

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

4.3.7.3 Tolerance

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

Arithmetic precision shall be tested according to the level definition.

4.3.8 Miscellaneous Conformance

4.3.8.1 Conformance Requirements

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

4.3.8.1.1 BIFS acquisition

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

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

4.3.8.1.2 Handling of discontinuities

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

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

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

4.3.8.1.3 Private data handling

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

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

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

4.3.8.2 Measurement Procedure

4.3.8.2.1 Buffer overflow/underflow tests

Continuous OTB shall be present and available.

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

4.4 Test material and test suites

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

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

The test sequences are bundled with an HTML file describing:

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

Some test sequences may also be accompanied by:

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

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

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

4.4.1 Parsing Hint File Format

4.4.1.1 Requirements

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

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

4.4.1.2 Syntax Elements in a Log File

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

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

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

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

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

Table 2 — Interpretation of syntax elements in a log file

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

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

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

Table 3 — Values for <TOS>

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

4.4.1.2.1 <semantic> syntax element

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

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

4.4.1.2.2 <decoded value> syntax element

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

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

Therefore, this field shall be left open for additional but non-normative information, which should provide some useful information for human readers¹. Guidelines for the provision of useful information by this syntax element are given in the following subclause.

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

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

4.4.1.3 Example

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

BIFS 5 (1, 3) isReused 0 FALSE

This line corresponds to the following information:

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

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

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

4.4.1.4 Suffix for Log Files

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

4.4.2 Scene Dump File Format

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

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

4.4.2.1 Elements and their attributes

4.4.2.1.1 <scene-dump>

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

Container:

The file.

Attributes:

version-number Required. Set to "1.0"

4.4.2.1.2 <timestamp>

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

Container:

<scene-dump>

Attributes:

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

4.4.2.1.3 <scene>

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

Container:

<scene-dump>

Attributes:

(none)

4.4.2.1.4 <node>

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

Container:

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

Attributes:

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

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

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

name Optional. Contains the name of the node.

4.4.2.1.5 <field>

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

Container:

<node>

Attributes:

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

name Optional. Contains the name of the field.

4.4.2.1.6 <indexed-value>

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

Container:

<field>

Attributes:

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

4.4.2.1.7 <routes>

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

Container:

<scene-dump>

Attributes:

(none)

4.4.2.1.8 <route>

This element describes a route in the scene.

Container:

<routes>

Attributes:

id Required. Contains the route's id.

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

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

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

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

4.4.2.2 Example file

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

```

</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>
  </node>
</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</indexed-value>
    </field>
  </node>
</indexed-value>
</field>
</node>
</scene>
<routes>
  <route id="1" src-node="3" src-field="6" dst-node="4" dst-field="0"/>
  <route id="2" src-node="4" src-field="2" dst-node="2" dst-field="4"/>
</routes>
</scene-dump>

```

4.4.3 Test Suites

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

4.4.3.1 BIFS Feature List

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

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

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

Table 4 — BIFS Test Suite Information

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

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

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

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

N°	Feature	Reference of Test sequence and associated method
117.	FontStyle	Scaling3D, Ecran2
118.	Form	Form_spread, Form_spread2, Testform
119.	Group	Anchor, Ecran2, Layout
120.	ImageTexture	Ecran2, Jerusalem, Pae_raise
121.	IndexedFaceSet	Test2, Test3
122.	IndexedFaceSet2D	lfs
123.	IndexedLineSet	
124.	IndexedLineSet2D	Polygontest, Updatetest, Mosaic18
125.	Inline	
126.	LOD	
127.	Layer2D	Bifs-2dfieldreplace1, Transition
128.	Layer3D	Bifs-deletenode, Scaling3D
129.	Layout	Jerusalem, Layout, Testlayout
130.	LineProperties	Ecran2, Updatetest
131.	ListeningPoint	
132.	Material	Bifs-deletenode, Material3D
133.	Material2D	Bifs-2dfieldreplace1, Ecran2
134.	MovieTexture	Jerusalem, Friday, Av
135.	NavigationInfo	nist-enst/Bindable_Nodes/NavigationInfo/*
136.	Normal	nist-enst/Geometric_Properties/Normal/*
137.	NormalInterpolator	
138.	OrderedGroup	Form_spread2, Pae_raise
139.	OrientationInterpolator	OrientInterp3D
140.	PixelTexture	
141.	PlaneSensor	
142.	PlaneSensor2D	Slider, Valuator
143.	PointLight	PointLightPrimitive1-3D
144.	PointSet	
145.	PointSet2D	PointSet2D, MovingPointFish2D
146.	PositionInterpolator	Value_changed3d
147.	PositionInterpolator2D	Friday, Traj0
148.	ProximitySensor2D	ProxSensInterp2D, ProximitysensorSimple2D
149.	ProximitySensor	
150.	QuantizationParameter	QuantQPtest, QuantDefUse
151.	Rectangle	Ecran2, Updatetest, Friday
152.	ScalarInterpolator	Trans-group
153.	Script	Scaling3D, SFCOLOR01, Value_changed3d, Qtvr
154.	Shape	Bifs-deletenode, Ecran2, Jerusalem
155.	Sound	
156.	Sound2D	Audiotest, lfs
157.	Sphere	Bifs-InsertNodeStress
158.	SphereSensor	

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

4.4.3.2 OD Feature List

Table 5 — OD Test Suite Information

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

4.4.3.3 Bitstreams

Table 6 – Test Sequence Providers and Reason for Existence

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

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

4.5 Advanced BIFS

4.5.1 Bitstream conformance

4.5.1.1 Conformance Requirements

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

4.5.1.2 Measurement procedure

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

4.5.1.3 Tolerance

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

4.5.2 Terminal conformance

4.5.2.1 Conformance Requirements

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

4.5.2.2 Measurement Procedure

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

4.5.2.3 Tolerance

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

4.6 MPEG-J

4.6.1 MPEG-J Conformance Points

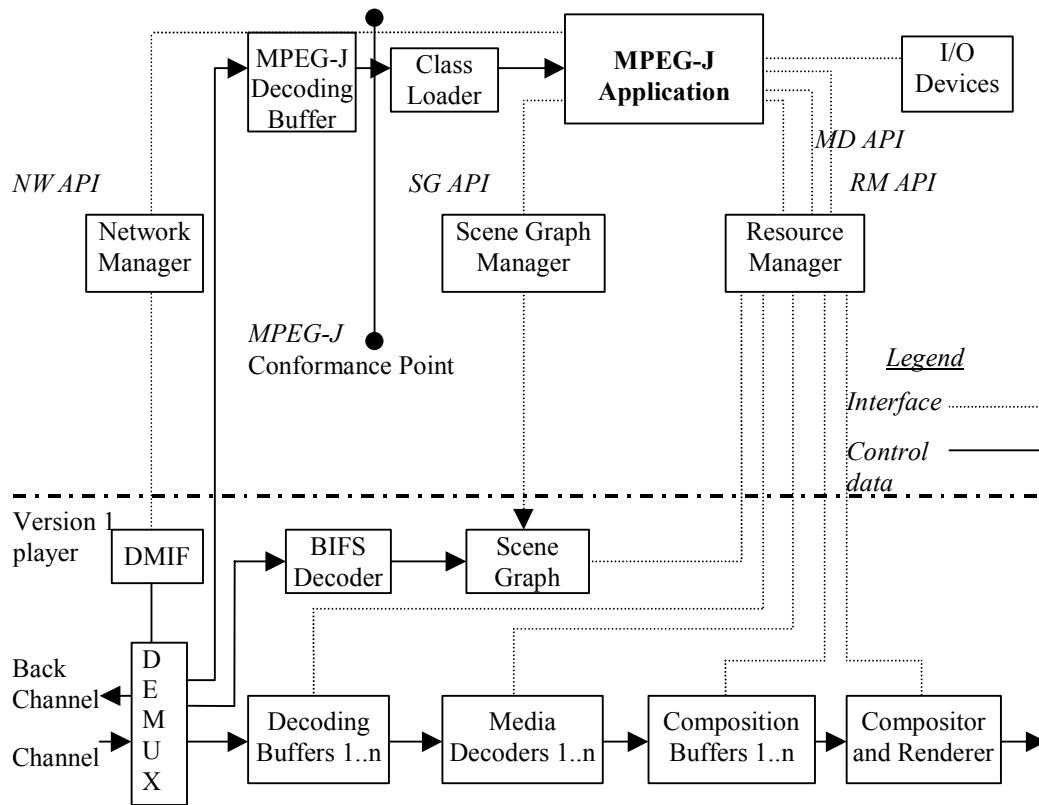


Figure 2 — MPEG-J Architecture with Conformance Point

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

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

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

Bitstream conformance point for MPEG-J is:

- MPEG-J Decoding

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

Terminal conformance point for MPEG-J is:

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

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

4.6.2 Bitstream Conformance

Each bitstream shall meet the syntactic and semantic requirements specified in ISO/IEC 14496-1. This subclause describes a set of tests to be performed on bitstreams. In the description of the tests it is assumed that the tested bitstream contains no errors due to transmission or other causes. For each test the condition or conditions that must be satisfied are given, as well as the prerequisites or conditions in which the test can be applied. Note that the application of these tests requires parsing of the bitstream to the appropriate levels. Parsing and interpretation of ODs is also required. In some cases of IPMP-protected data, de-scrambling may be required before the tests can be performed on non IPMP-related features.

4.6.2.1 MPEG-J Conformance

4.6.2.1.1 Conformance Requirements

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

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

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

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

4.6.2.1.2 Measurement procedure

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

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

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

Method `package_name.class_name.method_name` with parameter `parameter1` `parameter2` `parameter3...` `parametern` was called

where: *method_name* is the name of the method, *parameter_n* is:

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

E.g. a method `foo(var1, var2)` would print the trace

```
Method org.iso.mpeg.mpegj.foo with parameter var1 var2
Exception package_name.exception_name was thrown ( or )
Exception package_name.exception_name was thrown or with parameter var1
```

4.6.2.1.3 Tolerance

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

4.6.3 Terminal Conformance

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

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

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

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

4.6.3.1 MPEG-J conformance

4.6.3.1.1 Conformance Requirements

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

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

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

4.6.3.1.1.1 MPEG-J Decoding

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

- a. Recovering the access unit data (class, object, or zip file) from the bit stream. This is input to the MPEG-J Class Loader.
- b. Loading:
 - If the data is a class file it is loaded according to the rules specified in subclause 11.4 of ISO/IEC 14496-1.
 - If the data is a zip file the classes specified in the header are loaded according to the rules specified in subclause 11.4 of ISO/IEC 14496-1.
 - If the data is neither a class or a zip file, it is made available according to the rules specified in subclause 11.4 of ISO/IEC 14496-1.

4.6.3.1.1.2 MPEG-J API conformance

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

4.6.3.1.1.3 Java Platform conformance

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

4.6.3.1.2 Measurement procedure

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

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

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

4.6.3.1.3 Tolerance

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

4.7 MP4 File Format

4.7.1 Writing

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

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

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

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

Extensions should use the UUID mechanism.

Track identifiers must be unique within the file.

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

Fields marked reserved should be written to the standard value.

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

4.7.2 Reading

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

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

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

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

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

5 Visual

5.1 Introduction

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

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

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

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

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

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

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

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

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

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

5.2 Definition of visual bitstream compliance

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

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

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

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

5.2.1 Requirements and restrictions related to profile-and-level

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

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

5.2.2 Additional restrictions on bitstream applied by the encoder

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

5.2.3 Encoder requirements and recommendations

5.2.3.1 Encoder requirements

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

- 1) The bitstreams generated by the encoder are compliant *Profile@Level* bitstreams.
- 2) For encoding methods which include embedded decoding operations to produce the coded bitstream, these decoding operations shall be performed with the full arithmetic precision specified in ISO/IEC 14496-2.

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

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

5.2.3.2 Encoder recommendations

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

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

5.2.3.3 Restrictions on the Operation of the Reference Software

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

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

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

5.3 Procedure for testing bitstream compliance

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

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

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

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

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

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

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

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

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

- It allows to test the subjective quality of the reconstructed frames. ISO/IEC 14496-2 does not put any requirement on subjective quality, but it is desirable that an encoder generates bitstreams for which the subjective quality of reconstructed frames is as good as possible.
- Checking the output of the IDCT can provide an indication of whether or not the encoder that produced the bitstream observed the recommendation of the note in subclause 7.4.4 of ISO/IEC 14496-2.

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

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

5.4 Definition of visual decoder compliance

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

- 1) In item (1) of subclause 3.2 of the IEEE specification, the last sentence is replaced by: <<Data sets of 1 000 000 (one million) blocks each should be generated for (L=256, H=255), (L=H=5) and (L=384, H=383). >>
- 2) The text of subclause 3.3 of the IEEE specification is replaced by : <<For any pixel location, the peak error shall not exceed 2 in magnitude. There is no other accuracy requirement for this test.>>
- 3) Let F be the set of 4096 blocks $Bi[y][x]$ ($i=0..4095$) defined as follows :
 - a) $Bi[0][0] = i - 2048$
 - b) $Bi[7][7] = 1$ if $Bi[0][0]$ is even, $Bi[7][7] = 0$ if $Bi[0][0]$ is odd
 - c) All other coefficients $Bi[y][x]$ other than $Bi[0][0]$ and $Bi[7][7]$ are equal to 0

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

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

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

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

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

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

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

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

5.4.5 Requirement on output of the decoding process and timing

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

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

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

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

5.4.6 Recommendations

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

This subclause lists some of the recommendations.

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

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

5.5 Procedure to test decoder compliance

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

5.5.1 Static tests

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

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

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

There are two sorts of static tests.

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

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

5.5.2 Dynamic tests

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

5.5.3 Specification of the test bitstreams

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

- a) General
- b) Shape coding
- c) Scalability
- d) Error resilience
- e) Scalable still texture
- f) Sprites

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

1. General static tests:

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

2. Memory bandwidth dynamic tests:

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

3. VLC/FLC decoding static tests:

Bitstreams using all the possible events within a table.

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

Bitstreams containing very irregular distribution of bits or symbols.

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

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

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

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

5.5.3.1 Test Bitstreams – General

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

5.5.3.1.1 Test bitstream #GE-1

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

Functional stage: prediction bandwidth

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

5.5.3.1.2 Test bitstream #GE-2

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

Functional stage: VLD

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

5.5.3.1.3 Test bitstream #GE-3

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

Functional stage: prediction bandwidth

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

5.5.3.1.4 Test bitstream #GE-4

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

Functional stage: parser

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

5.5.3.1.5 Test bitstream #GE-6

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

Functional stage: parser and control

Purpose: Check that decoder handle all scenarios.

5.5.3.1.6 Test bitstream #GE-8

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

Functional stage: parser ; IDCT accuracy

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

5.5.3.1.7 Test bitstream #GE-9

This test has been removed from the test suite specification.

5.5.3.1.8 Test bitstream #GE-10

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

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

Functional stage: MCP

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

5.5.3.1.9 Test bitstream #GE-11

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

Functional stage: VLD

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

5.5.3.1.10 Test bitstream #GE-12

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

Functional stage: VLD and prediction bandwidth

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

5.5.3.1.11 Test bitstream #GE-13

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

Functional stage: prediction bandwidth

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

5.5.3.1.12 Test bitstream #GE-14

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

Functional stage: prediction bandwidth

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

5.5.3.1.13 Test bitstream #GE-16

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

Functional stage: overall

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

5.5.3.1.14 Test bitstream #GE-18

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

Functional stage: controller

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

5.5.3.1.15 Test bitstream #GE-19

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

Functional stage: IDCT

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

5.5.3.1.16 Test bitstream #GE-20

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

Functional stage: inverse quantization

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

5.5.3.1.17 Test bitstream #GE-21

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

Functional stage: inverse quantization

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

5.5.3.1.18 Test bitstream #GE-22

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

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

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

5.5.3.1.19 Test bitstream #GE-23

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

Functional stage: reconstruction of motion vectors, MCP, control

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

5.5.3.1.20 Test bitstream #GE-24

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

Functional stage: quantizer matrix download, matrix scanning.

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

5.5.3.1.21 Test bitstream #GE-25

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

Functional stage: IDCT

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

5.5.3.1.22 Test Bitstream #MHH-1

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

Functional stage: Motion vector decoding; motion compensation.

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

5.5.3.1.23 Test Bitstream #MHH-2

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

Functional stage: Motion vector decoding; motion compensation.

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

5.5.3.1.24 Test Bitstream #MHH-3

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

Functional stage: Motion vector decoding; motion compensation.

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

5.5.3.1.25 Test Bitstream #MHH-4

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

Functional stage: Motion vector decoding; motion compensation.

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

5.5.3.1.26 Test Bitstream #MHH-5

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

Functional stage: Motion vector decoding; motion compensation.

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

5.5.3.1.27 Test Bitstream #MHH-6

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

Functional stage: Motion vector decoding; motion compensation.

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

5.5.3.1.28 Test Bitstream #MHH-7

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

Functional stage: Motion vector decoding; motion compensation.

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

5.5.3.1.29 Test Bitstream #MVH-1

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

Functional stage: Motion vector decoding; motion compensation.

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

5.5.3.1.30 Test Bitstream #MVH-2

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

Functional stage: Motion vector decoding; motion compensation.

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

5.5.3.1.31 Test Bitstream #MVH-3

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

Functional stage: Motion vector decoding; motion compensation.

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

5.5.3.1.32 Test Bitstream #MVH-4

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

Functional stage: Motion vector decoding; motion compensation.

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

5.5.3.1.33 Test Bitstream #MVH-5

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

Functional stage: Motion vector decoding; motion compensation.

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

5.5.3.1.34 Test Bitstream #MVH-6

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

Functional stage: Motion vector decoding; motion compensation.

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