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

**Part 1:
Systems**

*Technologies de l'information — Codage des objets audiovisuels —
Partie 1: Systèmes*

Reference number
ISO/IEC 14496-1:1999(E)



PDF disclaimer

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

Adobe is a trademark of Adobe Systems Incorporated.

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

© ISO/IEC 1999

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 22 734 10 79
E-mail copyright@iso.ch
Web www.iso.ch

Printed in Switzerland

Contents	Page
0	Introduction xix
0.1	Overview xix
0.2	Architecture xix
0.3	Terminal Model: Systems Decoder Model.....xxi
0.3.1	Timing Model xxi
0.3.2	Buffer Model xxi
0.4	Multiplexing of Streams: The Delivery Layer xxi
0.5	Synchronization of Streams: The Sync Layer.....xxi
0.6	The Compression Layer xxii
0.6.1	Object Description Framework.....xxii
0.6.2	Scene Description Streams.....xxii
0.6.3	Audio-visual Streams xxiii
0.6.4	Upchannel Streams.....xxiii
1	Scope..... 1
2	Normative References 1
3	Additional References 2
4	Definitions..... 2
5	Abbreviations and Symbols..... 6
6	Conventions 7
7	Systems Decoder Model..... 7
7.1	Introduction 7
7.2	Concepts of the Systems Decoder Model 7
7.2.1	DMIF Application Interface (DAI) 7
7.2.2	SL-Packetized Stream (SPS)..... 8
7.2.3	Access Units (AU) 8
7.2.4	Decoding Buffer (DB)..... 8
7.2.5	Elementary Streams (ES) 8

7.2.6	Elementary Stream Interface (ESI)	8
7.2.7	Decoder	8
7.2.8	Composition Units (CU).....	8
7.2.9	Composition Memory (CM)	8
7.2.10	Compositor	9
7.3	Timing Model Specification.....	9
7.3.1	System Time Base (STB).....	9
7.3.2	Object Time Base (OTB)	9
7.3.3	Object Clock Reference (OCR)	9
7.3.4	Decoding Time Stamp (DTS).....	9
7.3.5	Composition Time Stamp (CTS)	10
7.3.6	Occurrence and Precision of Timing Information in Elementary Streams	10
7.3.7	Time Stamps for Dependent Elementary Streams.....	10
7.4	Buffer Model Specification.....	11
7.4.1	Elementary Decoder Model	11
7.4.2	Assumptions.....	11
7.4.3	Managing Buffers: A Walkthrough	12
8	Object Description Framework	13
8.1	Introduction	13
8.2	Common data structures.....	14
8.2.1	Overview	14
8.2.2	BaseDescriptor.....	15
8.2.3	BaseCommand	16
8.3	Intellectual Property Management and Protection (IPMP).....	16
8.3.1	Overview	16
8.3.2	IPMP Streams	17
8.4	Object Content Information (OCI).....	18
8.4.1	Overview	18
8.4.2	OCI Streams.....	18
8.5	Object Descriptor Stream.....	19

8.5.1	Structure of the Object Descriptor Stream.....	19
8.5.2	Access Unit Definition	19
8.5.3	Time Base for Object Descriptor Streams.....	20
8.5.4	OD Decoder Configuration.....	20
8.5.5	OD Command Syntax and Semantics.....	20
8.6	Object Descriptor Components	22
8.6.1	Overview	22
8.6.2	ObjectDescriptor	22
8.6.3	InitialObjectDescriptor.....	23
8.6.4	ES_Descriptor	26
8.6.5	DecoderConfigDescriptor	28
8.6.6	DecoderSpecificInfo	29
8.6.7	SLConfigDescriptor	30
8.6.8	IP_IdentificationDataSet	30
8.6.9	ContentIdentificationDescriptor	30
8.6.10	SupplementaryContentIdentificationDescriptor	32
8.6.11	IPI_DescrPointer	32
8.6.12	IPMP_DescriptorPointer.....	33
8.6.13	IPMP Descriptor	33
8.6.14	QoS_Descriptor	34
8.6.15	ExtensionDescriptor	35
8.6.16	RegistrationDescriptor	35
8.6.17	Object Content Information Descriptors	36
8.7	Rules for Usage of the Object Description Framework	41
8.7.1	Aggregation of Elementary Stream Descriptors in a Single Object Descriptor	41
8.7.2	Linking Scene Description and Object Descriptors	43
8.7.3	ISO/IEC 14496 Content Access.....	44
8.8	Usage of the IPMP System interface.....	50
8.8.1	Overview	50
8.8.2	Association of an IPMP System with ISO/IEC 14496 content.....	50

8.8.3	IPMP of Object Descriptor streams	51
8.8.4	IPMP of Scene Description streams.....	51
8.8.5	Usage of URLs in managed and protected content.....	51
8.8.6	IPMP Decoding Process	52
9	Scene Description	53
9.1	Introduction	53
9.1.1	Scope.....	53
9.1.2	Composition and Rendering	54
9.1.3	Scene Description	54
9.2	Concepts	55
9.2.1	BIFS Elementary Streams	55
9.2.2	BIFS Scene Graph	57
9.2.3	Sources of modification to the scene	64
9.3	BIFS Syntax	67
9.3.1	Introduction	67
9.3.2	Decoding tables, data structures and associated functions.....	68
9.3.3	Quantization.....	73
9.3.4	Compensation process	82
9.3.5	BIFS Configuration.....	83
9.3.6	BIFS Command Syntax.....	87
9.3.7	BIFS Scene.....	94
9.3.8	BIFS-Anim	115
9.4	Node Semantics	121
9.4.1	Overview	121
9.4.2	Node specifications	121
10	Synchronization of Elementary Streams	192
10.1	Introduction	192
10.2	Sync Layer	192
10.2.1	Overview	192
10.2.2	SL Packet Specification.....	193

10.2.3	SL Packet Header Configuration	193
10.2.4	SL Packet Header Specification	196
10.2.5	Clock Reference Stream.....	199
10.2.6	Restrictions for elementary streams sharing the same object time base	199
10.2.7	Usage of configuration options for object clock reference and time stamp values	200
10.3	Elementary Stream Interface (Informative).....	201
10.4	DMIF Application Interface.....	203
11	Multiplexing of Elementary Streams	203
11.1	Introduction	203
11.2	FlexMux Tool	203
11.2.1	Overview	203
11.2.2	Simple Mode	203
11.2.3	MuxCode mode	204
11.2.4	FlexMux packet specification	204
11.2.5	Usage of MuxCode Mode	206
12	Syntactic Description Language	207
12.1	Introduction	207
12.2	Elementary Data Types.....	207
12.2.1	Constant-Length Direct Representation Bit Fields	207
12.2.2	Variable Length Direct Representation Bit Fields	208
12.2.3	Constant-Length Indirect Representation Bit Fields.....	208
12.2.4	Variable Length Indirect Representation Bit Fields	209
12.3	Composite Data Types	210
12.3.1	Classes.....	210
12.3.2	Abstract Classes	211
12.3.3	Expandable classes	211
12.3.4	Parameter types	212
12.3.5	Arrays.....	212
12.3.6	Partial Arrays.....	213
12.3.7	Implicit Arrays	213

12.4	Arithmetic and Logical Expressions	214
12.5	Non-Parsable Variables	214
12.6	Syntactic Flow Control	214
12.7	Built-In Operators.....	216
12.8	Scoping Rules	216
13	Profiles	216
13.1	Introduction	216
13.2	OD Profile Definitions	216
13.2.1	Overview	216
13.2.2	OD Profiles Tools	216
13.2.3	OD Profiles.....	217
13.2.4	OD Profiles@Levels	217
13.3	Scene Graph Profile Definitions	217
13.3.1	Overview	217
13.3.2	Scene Graph Profiles Tools	217
13.3.3	Scene Graph Profiles	217
13.3.4	Scene Graph Profiles@Levels	220
13.4	Graphics Profile Definitions.....	221
13.4.1	Overview	221
13.4.2	Graphics Profiles Tools	221
13.4.3	Graphics Profiles.....	221
13.4.4	Graphics Profiles@Levels.....	222
Annex A (informative)	Bibliography	225
Annex B (informative)	Time Base Reconstruction.....	226
B.1	Time Base Reconstruction.....	226
B.1.1	Adjusting the Receiving Terminal's OTB.....	226
B.1.2	Mapping Time Stamps to the STB	226
B.1.3	Adjusting the STB to an OTB.....	227
B.1.4	System Operation without Object Time Base	227
B.2	Temporal aliasing and audio resampling	227

B.3	Reconstruction of a Synchronised Audio-visual Scene: A Walkthrough	227
Annex C (normative) View Dependent Object Scalability		229
C.1	Introduction	229
C.2	Bitstream Syntax	229
C.2.1	View Dependent Object	229
C.3	Bitstream Semantics	230
C.3.1	View Dependent Object	230
C.3.2	View Dependent Object Layer	230
Annex D (informative) Registration procedure		232
D.1	Procedure for the request of a Registration ID (RID)	232
D.2	Responsibilities of the Registration Authority	232
D.3	Contact information for the Registration Authority	232
D.4	Responsibilities of Parties Requesting a RID	232
D.5	Appeal Procedure for Denied Applications	233
D.6	Registration Application Form	233
D.6.1	Contact Information of organization requesting a RID	233
D.6.2	Request for a specific RID	233
D.6.3	Short description of RID that is in use and date system was implemented	233
D.6.4	Statement of an intention to apply the assigned RID	233
D.6.5	Date of intended implementation of the RID	234
D.6.6	Authorized representative	234
D.6.7	For official use of the Registration Authority	234
Annex E (informative) The QoS Management Model for ISO/IEC 14496 Content		235
Annex F (informative) Conversion Between Time and Date Conventions		236
Annex G (normative) Adaptive Arithmetic Decoder for BIFS-Anim		238
Annex H (normative) Node coding tables		240
H.1	Node Tables	240
H.1.1	Anchor	240
H.1.2	AnimationStream	240
H.1.3	Appearance	241

H.1.4	AudioBuffer.....	241
H.1.5	AudioClip	241
H.1.6	AudioDelay.....	241
H.1.7	AudioFX.....	241
H.1.8	AudioMix	242
H.1.9	AudioSource	242
H.1.10	AudioSwitch.....	242
H.1.11	Background	242
H.1.12	Background2D.....	243
H.1.13	Billboard.....	243
H.1.14	Bitmap	243
H.1.15	Box.....	243
H.1.16	Circle	243
H.1.17	Collision	243
H.1.18	Color	244
H.1.19	ColorInterpolator	244
H.1.20	CompositeTexture2D	244
H.1.21	CompositeTexture3D	244
H.1.22	Conditional.....	244
H.1.23	Cone	245
H.1.24	Coordinate	245
H.1.25	Coordinate2D.....	245
H.1.26	CoordinateInterpolator	245
H.1.27	CoordinateInterpolator2D.....	245
H.1.28	Curve2D.....	245
H.1.29	Cylinder	246
H.1.30	CylinderSensor.....	246
H.1.31	DirectionalLight	246
H.1.32	DiscSensor.....	246
H.1.33	ElevationGrid	247

H.1.34	Expression	247
H.1.35	Extrusion	247
H.1.36	Face	248
H.1.37	FaceDefMesh	248
H.1.38	FaceDefTables	248
H.1.39	FaceDefTransform	248
H.1.40	FAP	248
H.1.41	FDP	250
H.1.42	FIT	250
H.1.43	Fog	250
H.1.44	FontStyle	250
H.1.45	Form	251
H.1.46	Group	251
H.1.47	ImageTexture	251
H.1.48	IndexedFaceSet	251
H.1.49	IndexedFaceSet2D	252
H.1.50	IndexedLineSet	252
H.1.51	IndexedLineSet2D	252
H.1.52	Inline	252
H.1.53	LOD	253
H.1.54	Layer2D	253
H.1.55	Layer3D	253
H.1.56	Layout	253
H.1.57	LineProperties	254
H.1.58	ListeningPoint	254
H.1.59	Material	254
H.1.60	Material2D	254
H.1.61	MovieTexture	254
H.1.62	NavigationInfo	255
H.1.63	Normal	255

H.1.64	NormalInterpolator	255
H.1.65	OrderedGroup.....	255
H.1.66	OrientationInterpolator	255
H.1.67	PixelTexture	255
H.1.68	PlaneSensor.....	256
H.1.69	PlaneSensor2D	256
H.1.70	PointLight.....	256
H.1.71	PointSet.....	256
H.1.72	PointSet2D	256
H.1.73	PositionInterpolator	257
H.1.74	PositionInterpolator2D.....	257
H.1.75	ProximitySensor2D	257
H.1.76	ProximitySensor.....	257
H.1.77	QuantizationParameter	257
H.1.78	Rectangle	258
H.1.79	ScalarInterpolator.....	258
H.1.80	Script	259
H.1.81	Shape.....	259
H.1.82	Sound	259
H.1.83	Sound2D.....	259
H.1.84	Sphere	259
H.1.85	SphereSensor	259
H.1.86	SpotLight.....	260
H.1.87	Switch.....	260
H.1.88	TermCap.....	260
H.1.89	Text	260
H.1.90	TextureCoordinate	260
H.1.91	TextureTransform.....	261
H.1.92	TimeSensor.....	261
H.1.93	TouchSensor	261

H.1.94	Transform.....	261
H.1.95	Transform2D.....	261
H.1.96	Valuator.....	262
H.1.97	Viewpoint	263
H.1.98	VisibilitySensor	263
H.1.99	Viseme.....	263
H.1.100	WorldInfo	263
H.2	Node Definition Type Tables.....	263
H.2.1	SF2DNode	263
H.2.2	SF3DNode	264
H.2.3	SFAppearanceNode	265
H.2.4	SFAudioNode	265
H.2.5	SFBackground2DNode	265
H.2.6	SFBackground3DNode	265
H.2.7	SFColorNode	266
H.2.8	SFCoordinate2DNode	266
H.2.9	SFCoordinateNode.....	266
H.2.10	SFExpressionNode	266
H.2.11	SFFAPNode	266
H.2.12	SFFDPNode	266
H.2.13	SFFITNode	266
H.2.14	SFFaceDefMeshNode	266
H.2.15	SFFaceDefTablesNode	266
H.2.16	SFFaceDefTransformNode.....	266
H.2.17	SFFogNode	267
H.2.18	SFFontStyleNode	267
H.2.19	SFGeometryNode.....	267
H.2.20	SFLinePropertiesNode	267
H.2.21	SFMaterialNode	267
H.2.22	SFNavigationInfoNode.....	267

H.2.23	SFNormalNode	267
H.2.24	SFStreamingNode	268
H.2.25	SFTextureCoordinateNode.....	268
H.2.26	SFTextureNode.....	268
H.2.27	SFTextureTransformNode.....	268
H.2.28	SFTopNode	268
H.2.29	SFViewpointNode.....	268
H.2.30	SFVisemeNode	268
H.2.31	SFWorldNode.....	268
Annex I (informative) MPEG-4 Audio TTS application with Facial Animation		271
Annex J (informative) Graphical representation of object descriptor and sync layer syntax.....		272
J.1	Length encoding of descriptors and commands.....	272
J.2	Object Descriptor Stream and OD commands.....	272
J.3	IPMP stream.....	273
J.4	OCI stream	273
J.5	Object descriptor and its components	273
J.6	OCI Descriptors	275
J.7	Sync layer configuration and syntax	278
Annex K (informative) Patent statements		280

Figures

Figure 1 - The ISO/IEC 14496 terminal architecture	xx
Figure 2 - Systems Decoder Model	7
Figure 3 - Composition unit availability.....	10
Figure 4 - Flow diagram for the Systems Decoder Model	11
Figure 5 - Object descriptors linking scene description to elementary streams.....	14
Figure 6 - Complex content example	48
Figure 7 - Requesting stream delivery through the DAI	50
Figure 8 - IPMP system in the ISO/IEC 14496 terminal architecture.....	52

Figure 9 - An example of an object-based multimedia scene	53
Figure 10 - Logical structure of example scene	54
Figure 11 - Media start times and CTS.....	57
Figure 12 - Scene graph example.....	58
Figure 13 - 2D co-ordinate system (AR = Aspect Ratio)	59
Figure 14 - BIFS-Command Types	66
Figure 15 - A CompositeTexture2D example. The 2D scene is projected onto the 3D cube.	134
Figure 16 - A CompositeTexture2D example.	134
Figure 17 - CompositeTexture3D example. The 3D view of the earth is projected onto the 3D cube	135
Figure 18 - Curve node example	140
Figure 19 - An arbitrary motion trajectory is approximated as a piece-wise linear one.	144
Figure 20 - A FIG example.....	151
Figure 21 - Visual result of the Form node example	157
Figure 22 - IndexedFaceSet2D default texture mapping coordinates for a simple shape	160
Figure 23 - Three Layer2D and Layer3D examples composed in a 2D space.....	164
Figure 24 - Cap and join style for LineProperties	167
Figure 25 - Valuator functionaliy	189
Figure 26 - The sync layer.....	192
Figure 27 - Structure of FlexMux packet in simple mode.....	204
Figure 28 - Structure of FlexMux packet in MuxCode mode	204
Figure 29 - Example for a FlexMux packet in MuxCode mode.....	206

Tables

Table 1 - List of Class Tags for Descriptors	15
Table 2 - List of Class Tags for Commands.....	16
Table 3 - ODProfileLevelIndication Values.....	24
Table 4 - sceneProfileLevelIndication Values.....	24
Table 5 - audioProfileLevelIndication Values	25
Table 6 - visualProfileLevelIndication Values.....	25

Table 7 - graphicsProfileLevelIndication Values	26
Table 8 - objectTypeIndication Values.....	28
Table 9 - streamType Values.....	29
Table 10 - contentType Values	31
Table 11 - contentIdentifierType Values	31
Table 12 - Predefined QoS Profiles	34
Table 13 - Standard units	62
Table 14 – Audio-Specific BIFS Nodes	64
Table 15 – Return values of getNbComp.....	71
Table 16 - Return values of getNbBounds	73
Table 17 - Quantization Categories	73
Table 18 - Condition for setting isQuantized to true.....	74
Table 19 - Value of nbBits depending on quantType	75
Table 20 - Value of floatMin, depending on quantType and fieldType.....	75
Table 21 - Value of floatMax, depending on quantType and fieldType.....	76
Table 22 - Value of intMin, depending on quantType	76
Table 23 – Animation Categories	77
Table 24 - Value of nbBits, depending on animType	77
Table 25 - Value of floatMin, depending on animType	78
Table 26 - Value of floatMax, depending on animType	78
Table 27 - Value of intMin, depending on animType	79
Table 28 - Quantization and inverse quantization process	79
Table 29 - Compensation process.....	83
Table 30 - Vertex displacements	144
Table 31 - Alignment Constraints.....	156
Table 32 - Distribution Constraints	156
Table 33 - lineStyle description.....	167
Table 34 - Semantics of value, dependent on capability	182
Table 35 - Semantics of value for capability=0	182
Table 36 - Semantics of value for capability=1	183

Table 37 - Semantics of value for capability=2	183
Table 38 - Semantics of value for capability=3	183
Table 39 - Semantics of value for capability=32	184
Table 40 - Semantics of value for capability=33	184
Table 41 - Semantics of value for capability=34	184
Table 42 - Semantics of value for capability=64	185
Table 43 - Semantics of value for capability=65	185
Table 44 - Simple typecasting conversion from other data types to float.	190
Table 45 - Simple typecasting conversion from float to other data types.	190
Table 46 - Overview of predefined SLConfigDescriptor values.....	194
Table 47 – Detailed predefined SLConfigDescriptor values.....	194
Table 48 – SLConfigDescriptor parameter values for a ClockReferenceStream	199
Table 49 - OD Profiles.....	217
Table 50 - Scene graph profiles.....	218
Table 51 - BIFS nodes for audio objects	219
Table 52 - BIFS nodes for visual objects.....	219
Table 53 - Restrictions for Simple 2D scene graph profile at Level 1	220
Table 54 - Graphics profiles.....	221

Foreword

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

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

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

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

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

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

- *Part 1: Systems*
- *Part 2: Visual*
- *Part 3: Audio*
- *Part 4: Conformance testing*
- *Part 5: Reference software*
- *Part 6: Delivery Multimedia Integration Framework (DMIF)*

Annexes C, G and H form a normative part of this part of ISO/IEC 14496. Annexes A, B, D, E, F, I, J and K are for information only.

0 Introduction

0.1 Overview

ISO/IEC 14496 specifies a system for the communication of interactive audio-visual scenes. This specification includes the following elements:

1. the coded representation of natural or synthetic, two-dimensional (2D) or three-dimensional (3D) objects that can be manifested audibly and/or visually (audio-visual objects) (specified in part 1,2 and 3 of ISO/IEC 14496);
2. the coded representation of the spatio-temporal positioning of audio-visual objects as well as their behavior in response to interaction (scene description, specified in this part of ISO/IEC 14496);
3. the coded representation of information related to the management of data streams (synchronization, identification, description and association of stream content, specified in this part of ISO/IEC 14496); and
4. a generic interface to the data stream delivery layer functionality (specified in part 6 of ISO/IEC 14496).

The overall operation of a system communicating audio-visual scenes can be paraphrased as follows:

At the sending terminal, the audio-visual scene information is compressed, supplemented with synchronization information and passed to a delivery layer that multiplexes it into one or more coded binary streams that are transmitted or stored. At the receiving terminal, these streams are demultiplexed and decompressed. The audio-visual objects are composed according to the scene description and synchronization information and presented to the end user. The end user may have the option to interact with this presentation. Interaction information can be processed locally or transmitted back to the sending terminal. ISO/IEC 14496 defines the syntax and semantics of the bitstreams that convey such scene information, as well as the details of their decoding processes.

This part of ISO/IEC 14496 specifies the following tools:

- a terminal model for time and buffer management;
- a coded representation of interactive audio-visual scene description information (Binary Format for Scenes – BIFS);
- a coded representation of metadata for the identification, description and logical dependencies of the elementary streams (Object descriptors and other Descriptors);
- a coded representation of descriptive audio-visual content information (object content information – OCI);
- an interface to intellectual property management and protection (IPMP) systems;
- a coded representation of synchronization information (sync layer – SL); and
- a multiplexed representation of individual elementary streams in a single stream (FlexMux).

These various elements are described functionally in this subclause and specified in the normative clauses that follow.

0.2 Architecture

The information representation specified in ISO/IEC 14496-1 describes the means to create an interactive audio-visual scene in terms of coded audio-visual information and associated scene description information. The entity that composes and sends, or receives and presents such a coded representation of an interactive audio-visual scene is generically referred to as an "audio-visual terminal" or just "terminal". This terminal may correspond to a standalone application or be part of an application system.

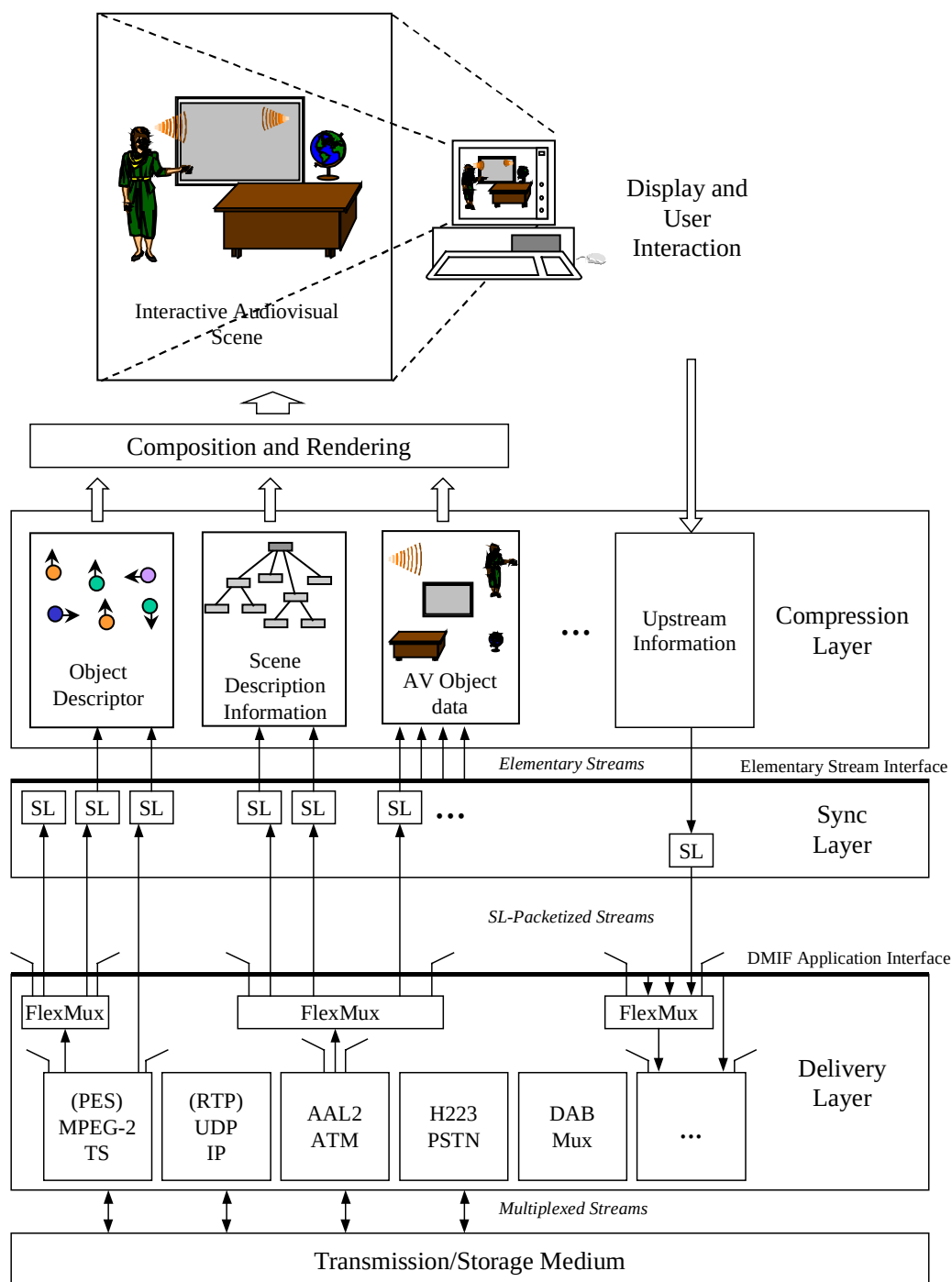


Figure 1 - The ISO/IEC 14496 terminal architecture

The basic operations performed by such a receiver terminal are as follows. Information that allows access to content complying with ISO/IEC 14496 is provided as initial session set up information to the terminal. Part 6 of ISO/IEC 14496 defines the procedures for establishing such session contexts as well as the interface to the delivery layer that generically abstracts the storage or transport medium. The initial set up information allows, in a recursive manner, to locate one or more elementary streams that are part of the coded content representation. Some of these elementary streams may be grouped together using the multiplexing tool described in ISO/IEC 14496-1.

Elementary streams contain the coded representation of either audio or visual data or scene description information. Elementary streams may as well themselves convey information to identify streams, to describe logical

dependencies between streams, or to describe information related to the content of the streams. Each elementary stream contains only one type of data.

Elementary streams are decoded using their respective stream-specific decoders. The audio-visual objects are composed according to the scene description information and presented by the terminal's presentation device(s). All these processes are synchronized according to the Systems Decoder Model (SDM) using the synchronization information provided at the synchronization layer.

These basic operations are depicted in Figure 1, and are described in more detail below.

0.3 Terminal Model: Systems Decoder Model

The systems decoder model provides an abstract view of the behavior of a terminal complying with ISO/IEC 14496-1. Its purpose is to enable a sending terminal to predict how the receiving terminal will behave in terms of buffer management and synchronization when reconstructing the audio-visual information that comprises the presentation. The systems decoder model includes a systems timing model and a systems buffer model which are described briefly in the following subclauses.

0.3.1 Timing Model

The timing model defines the mechanisms through which a receiving terminal establishes a notion of time that enables it to process time-dependent events. This model also allows the receiving terminal to establish mechanisms to maintain synchronization both across and within particular audio-visual objects as well as with user interaction events. In order to facilitate these functions at the receiving terminal, the timing model requires that the transmitted data streams contain implicit or explicit timing information. Two sets of timing information are defined in ISO/IEC 14496-1: clock references and time stamps. The former convey the sending terminal's time base to the receiving terminal, while the latter convey a notion of relative time for specific events such as the desired decoding or composition time for portions of the encoded audio-visual information.

0.3.2 Buffer Model

The buffer model enables the sending terminal to monitor and control the buffer resources that are needed to decode each elementary stream in a presentation. The required buffer resources are conveyed to the receiving terminal by means of descriptors at the beginning of the presentation. The terminal can then decide whether or not it is capable of handling this particular presentation. The buffer model allows the sending terminal to specify when information may be removed from these buffers and enables it to schedule data transmission so that the appropriate buffers at the receiving terminal do not overflow or underflow.

0.4 Multiplexing of Streams: The Delivery Layer

The term delivery layer is used as a generic abstraction of any existing transport protocol stack that may be used to transmit and/or store content complying with ISO/IEC 14496. The functionality of this layer is not within the scope of ISO/IEC 14496-1, and only the interface to this layer is considered. This interface is the DMIF Application Interface (DAI) specified in ISO/IEC 14496-6. The DAI defines not only an interface for the delivery of streaming data, but also for signaling information required for session and channel set up as well as tear down. A wide variety of delivery mechanisms exist below this interface, with some of them indicated in Figure 1. These mechanisms serve for transmission as well as storage of streaming data, i.e., a file is considered to be a particular instance of a delivery layer. For applications where the desired transport facility does not fully address the needs of a service according to the specifications in ISO/IEC 14496, a simple multiplexing tool (FlexMux) with low delay and low overhead is defined in ISO/IEC 14496-1.

0.5 Synchronization of Streams: The Sync Layer

Elementary streams are the basic abstraction for any streaming data source. Elementary streams are conveyed as sync layer-packetized (SL-packetized) streams at the DMIF Application Interface. This packetized representation additionally provides timing and synchronization information, as well as fragmentation and random access information. The sync layer (SL) extracts this timing information to enable synchronized decoding and, subsequently, composition of the elementary stream data.

0.6 The Compression Layer

The compression layer receives data in its encoded format and performs the necessary operations to decode this data. The decoded information is then used by the terminal's composition, rendering and presentation subsystems.

0.6.1 Object Description Framework

The purpose of the object description framework is to identify and describe elementary streams and to associate them appropriately to an audio-visual scene description. Object descriptors serve to gain access to ISO/IEC 14496 content. Object content information and the interface to intellectual property management and protection systems are also part of this framework.

An object descriptor is a collection of one or more elementary stream descriptors that provide the configuration and other information for the streams that relate to either an audio-visual object or a scene description. Object descriptors are themselves conveyed in elementary streams. Each object descriptor is assigned an identifier (object descriptor ID), which is unique within a defined name scope. This identifier is used to associate audio-visual objects in the scene description with a particular object descriptor, and thus the elementary streams related to that particular object.

Elementary stream descriptors include information about the source of the stream data, in form of a unique numeric identifier (the elementary stream ID) or a URL pointing to a remote source for the stream. Elementary stream descriptors also include information about the encoding format, configuration information for the decoding process and the sync layer packetization, as well as quality of service requirements for the transmission of the stream and intellectual property identification. Dependencies between streams can also be signaled within the elementary stream descriptors. This functionality may be used, for example, in scalable audio or visual object representations to indicate the logical dependency of a stream containing enhancement information, to a stream containing the base information. It can also be used to describe alternative representations for the same content (e.g. the same speech content in various languages).

0.6.1.1 Intellectual Property Management and Protection

The intellectual property management and protection (IPMP) framework for ISO/IEC 14496 content consists of a normative interface that permits an ISO/IEC 14496 terminal to host one or more IPMP Systems. The IPMP interface consists of IPMP elementary streams and IPMP descriptors. IPMP descriptors are carried as part of an object descriptor stream. IPMP elementary streams carry time variant IPMP information that can be associated to multiple object descriptors.

The IPMP System itself is a non-normative component that provides intellectual property management and protection functions for the terminal. The IPMP System uses the information carried by the IPMP elementary streams and descriptors to make protected ISO/IEC 14496 content available to the terminal. An application may choose not to use an IPMP System, thereby offering no management and protection features.

0.6.1.2 Object Content Information

Object content information (OCI) descriptors convey descriptive information about audio-visual objects. The main content descriptors are: content classification descriptors, keyword descriptors, rating descriptors, language descriptors, textual descriptors, and descriptors about the creation of the content. OCI descriptors can be included directly in the related object descriptor or elementary stream descriptor or, if it is time variant, it may be carried in an elementary stream by itself. An OCI stream is organized in a sequence of small, synchronized entities called events that contain a set of OCI descriptors. OCI streams can be associated to multiple object descriptors.

0.6.2 Scene Description Streams

Scene description addresses the organization of audio-visual objects in a scene, in terms of both spatial and temporal attributes. This information allows the composition and rendering of individual audio-visual objects after the respective decoders have reconstructed the streaming data for them. For visual data, ISO/IEC 14496-1 does not mandate particular composition algorithms. Hence, visual composition is implementation dependent. For audio data, the composition process is defined in a normative manner in 9.2.2.13 and ISO/IEC 14496-3.

The scene description is represented using a parametric approach (BIFS - Binary Format for Scenes). The description consists of an encoded hierarchy (tree) of nodes with attributes and other information (including event sources and targets). Leaf nodes in this tree correspond to elementary audio-visual data, whereas intermediate nodes group this material to form audio-visual objects, and perform grouping, transformation, and other such operations on audio-visual objects (scene description nodes). The scene description can evolve over time by using scene description updates.

In order to facilitate active user involvement with the presented audio-visual information, ISO/IEC 14496-1 provides support for user and object interactions. Interactivity mechanisms are integrated with the scene description information, in the form of linked event sources and targets (routes) as well as sensors (special nodes that can trigger events based on specific conditions). These event sources and targets are part of scene description nodes, and thus allow close coupling of dynamic and interactive behavior with the specific scene at hand. ISO/IEC 14496-1, however, does not specify a particular user interface or a mechanism that maps user actions (e.g., keyboard key presses or mouse movements) to such events.

Such an interactive environment may not need an upstream channel, but ISO/IEC 14496 also provides means for client-server interactive sessions with the ability to set up upstream elementary streams and associate them to specific downstream elementary streams.

0.6.3 Audio-visual Streams

The coded representations of audio and visual information are described in ISO/IEC 14496-3 and ISO/IEC 14496-2, respectively. The reconstructed audio-visual data are made available to the composition process for potential use during the scene rendering.

0.6.4 Upchannel Streams

Downchannel elementary streams may require upchannel information to be transmitted from the receiving terminal to the sending terminal (e.g., to allow for client-server interactivity). Figure 1 indicates the flowpath for an elementary stream from the receiving terminal to the sending terminal. The content of upchannel streams is specified in the same part of the specification that defines the content of the downstream data. For example, upchannel control streams for video downchannel elementary streams are defined in ISO/IEC 14496-2.

Information technology — Coding of audio-visual objects —

Part 1: Systems

1 Scope

This part of ISO/IEC 14496 specifies system level functionalities for the communication of interactive audio-visual scenes. More specifically:

1. system level description of the coded representation of natural or synthetic, two-dimensional (2D) or three-dimensional (3D) objects that can be manifested audibly and/or visually (audio-visual objects);
2. the coded representation of the spatio-temporal positioning of audio-visual objects as well as their behavior in response to interaction (scene description); and
3. the coded representation of information related to the management of data streams (synchronization, identification, description and association of stream content).

2 Normative References

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

- [1] ISO 639-2:1998, *Codes for the representation of names of languages – Part 2: Alpha-3 code*.
- [2] ISO 3166-1:1997, *Codes for the representation of names of countries and their subdivisions – Part 1: Country codes*.
- [3] ISO/IEC 10646-1:1993, *Information technology – Universal Multiple-Octet Coded Character Set (UCS) – Part 1: Architecture and Basic Multilingual Plane*.
- [4] 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*.
- [5] 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*.
- [6] 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*.
- [7] ITU-T Rec. H.262 (1995)||ISO/IEC 13818-2:1996, *Information technology – Generic coding of moving pictures and associated audio information: Video*.
- [8] ISO/IEC 13818-3:1998, *Information technology – Generic coding of moving pictures and associated audio information – Part 3: Audio*.

- [9] ISO/IEC 13818-7:1997, *Information technology – Generic coding of moving pictures and associated audio information – Part 7: Advanced Audio Coding (AAC)*.
- [10] ISO/IEC 14772-1:1998, *Information technology – Computer graphics and image processing – The Virtual Reality Modeling Language – Part 1: Functional specification and UTF-8 encoding*.
- [11] ISO/IEC 16262:1998, *Information technology – ECMAScript language specification*.
- [12] IEEE Std 754-1985, *Standard for Binary Floating-Point Arithmetic*.

3 Additional References

None cited.

4 Definitions

4.1 Access Unit (AU)

An individually accessible portion of data within an *elementary stream*. An access unit is the smallest data entity to which timing information can be attributed.

4.2 Alpha Map

The representation of the transparency parameters associated with a texture map.

4.3 Audio-visual Object

A representation of a natural or synthetic object that has an audio and/or visual manifestation. The representation corresponds to a node or a group of nodes in the BIFS scene description. Each audio-visual object is associated with zero or more *elementary streams* using one or more *object descriptors*.

4.4 Audio-visual Scene (AV Scene)

A set of audio-visual objects together with scene description information that defines their spatial and temporal attributes including behaviors resulting from object and user interactions.

4.5 Binary Format for Scene (BIFS)

A coded representation of a parametric scene description format.

4.6 Buffer Model

A model that defines how a terminal complying with ISO/IEC 14496 manages the buffer resources that are needed to decode a presentation.

4.7 Byte Aligned

A position in a coded bit stream with a distance of a multiple of 8-bits from the first bit in the stream.

4.8 Clock Reference

A special time stamp that conveys a reading of a time base.

4.9 Composition

The process of applying scene description information in order to identify the spatio-temporal attributes and hierarchies of audio-visual objects..

4.10 Composition Memory (CM)

A random access memory that contains composition units.

4.11 Composition Time Stamp (CTS)

An indication of the nominal composition time of a composition unit.

4.12 Composition Unit (CU)

An individually accessible portion of the output that a decoder produces from access units.

4.13 Compression Layer

The layer of a system according to the specifications in ISO/IEC 14496 that translates between the coded representation of an elementary stream and its decoded representation. It incorporates the decoders.

4.14 Decoder

An entity that translates between the coded representation of an elementary stream and its decoded representation.

4.15 Decoding buffer (DB)

A buffer at the input of a decoder that contains access units.

4.16 Decoder configuration

The configuration of a decoder for processing its elementary stream data by using information contained in its elementary stream descriptor.

4.17 Decoding Time Stamp (DTS)

An indication of the nominal decoding time of an access unit.

4.18 Delivery Layer

A generic abstraction for delivery mechanisms (computer networks, etc.) able to store or transmit a number of multiplexed elementary streams or FlexMux streams.

4.19 Descriptor

A data structure that is used to describe particular aspects of an elementary stream or a coded audio-visual object.

4.20 DMIF Application Interface (DAI)

An interface specified in ISO/IEC 14496-6. It is used here to model the exchange of SL-packetized stream data and associated control information between the sync layer and the delivery layer.

4.21 Elementary Stream (ES)

A consecutive flow of mono-media data from a single source entity to a single destination entity on the compression layer.

4.22 Elementary Stream Descriptor

A structure contained in object descriptors that describes the encoding format, initialization information, sync layer configuration, and other descriptive information about the content carried in an elementary stream.

4.23 Elementary Stream Interface (ESI)

An interface modeling the exchange of elementary stream data and associated control information between the compression layer and the sync layer.

4.24 FlexMux Channel (FMC)

A label to differentiate between data belonging to different constituent streams within one FlexMux Stream. A sequence of data in one FlexMux channel within a FlexMux stream corresponds to one single SL-packetized stream.

4.25 FlexMux Packet

The smallest data entity managed by the FlexMux tool. It consists of a header and a payload.

4.26 FlexMux Stream

A sequence of FlexMux Packets with data from one or more SL-packetized streams that are each identified by their own FlexMux channel.

4.27 FlexMux tool

A tool that allows the interleaving of data from multiple data streams.

4.28 Graphics Profile

A profile that specifies the permissible set of graphical elements of the BIFS tool that may be used in a scene description stream. Note that BIFS comprises both graphical and scene description elements.

4.29 Inter

A mode for coding parameters that uses previously coded parameters to construct a prediction.

4.30 Intra

A mode for coding parameters that does not make reference to previously coded parameters to perform the encoding.

4.31 Initial Object Descriptor

A special object descriptor that allows the receiving terminal to gain initial access to portions of content encoded according to ISO/IEC 14496. It conveys profile and level information to describe the complexity of the content.

4.32 Intellectual Property Identification (IPI)

A unique identification of one or more elementary streams corresponding to parts of one or more audio-visual objects.

4.33 Intellectual Property Management and Protection (IPMP) System

A generic term for mechanisms and tools to manage and protect intellectual property. Only the interface to such systems is normatively defined.

4.34 Object Clock Reference (OCR)

A clock reference that is used by a decoder to recover the time base of the encoder of an elementary stream.

4.35 Object Content Information (OCI)

Additional information about content conveyed through one or more elementary streams. It is either aggregated to individual elementary stream descriptors or is itself conveyed as an elementary stream.

4.36 Object Descriptor (OD)

A descriptor that aggregates one or more elementary streams by means of their elementary stream descriptors and defines their logical dependencies.

4.37 Object Descriptor Command

A command that identifies the action to be taken on a list of object descriptors or object descriptor IDs, e.g., update or remove.

4.38 Object Descriptor Profile

A profile that specifies the configurations of the object descriptor tool and the sync layer tool that are allowed.

4.39 Object Descriptor Stream

An elementary stream that conveys object descriptors encapsulated in object descriptor commands.

4.40 Object Time Base (OTB)

A time base valid for a given elementary stream, and hence for its decoder. The OTB is conveyed to the decoder via object clock references. All time stamps relating to this object's decoding process refer to this time base.

4.41 Parametric Audio Decoder

A set of tools for representing and decoding speech signals coded at bit rates between 6 Kbps and 16 Kbps, according to the specifications in ISO/IEC 14496-3.

4.42 Quality of Service (QoS)

The performance that an elementary stream requests from the delivery channel through which it is transported. QoS is characterized by a set of parameters (e.g., bit rate, delay jitter, bit error rate, etc.).

4.43 Random Access

The process of beginning to read and decode a coded representation at an arbitrary point within the elementary stream.

4.44 Reference Point

A location in the data or control flow of a system that has some defined characteristics.

4.45 Rendering

The action of transforming a scene description and its constituent audio-visual objects from a common representation space to a specific presentation device (i.e., speakers and a viewing window).

4.46 Rendering Area

The portion of the display device's screen into which the scene description and its constituent audio-visual objects are to be rendered.

4.47 Scene Description

Information that describes the spatio-temporal positioning of audio-visual objects as well as their behavior resulting from object and user interactions. The scene description makes reference to elementary streams with audio-visual data by means of pointers to object descriptors.

4.48 Scene Description Stream

An elementary stream that conveys scene description information.

4.49 Scene Graph Elements

The elements of the BIFS tool that relate only to the structure of the audio-visual scene (spatio-temporal positioning of audio-visual objects as well as their behavior resulting from object and user interactions) excluding the audio, visual and graphics nodes as specified in clause 13.

4.50 Scene Graph Profile

A profile that defines the permissible set of scene graph elements of the BIFS tool that may be used in a scene description stream. Note that BIFS comprises both graphical and scene description elements.

4.51 SL-Packetized Stream (SPS)

A sequence of sync layer Packets that encapsulate one elementary stream.

4.52 Structured Audio

A method of describing synthetic sound effects and music as defined by ISO/IEC 14496-3.

4.53 Sync Layer (SL)

A layer to adapt elementary stream data for communication across the DMIF Application Interface, providing timing and synchronization information, as well as fragmentation and random access information. The sync layer syntax is configurable and can be configured to be empty.

4.54 Sync Layer Configuration

A configuration of the sync layer syntax for a particular elementary stream using information contained in its elementary stream descriptor.

4.55 Sync Layer Packet (SL-Packet)

The smallest data entity managed by the sync layer consisting of a configurable header and a payload. The payload may consist of one complete access unit or a partial access unit.

4.56 Syntactic Description Language (SDL)

A language defined by ISO/IEC 14496-1 that allows the description of a bitstream's syntax.

4.57 Systems Decoder Model (SDM)

A model that provides an abstract view of the behavior of a terminal compliant to ISO/IEC 14496. It consists of the buffer model and the timing model.

4.58 System Time Base (STB)

The time base of the terminal. Its resolution is implementation-dependent. All operations in the terminal are performed according to this time base.

4.59 Terminal

A system that sends, or receives and presents the coded representation of an interactive audio-visual scene as defined by ISO/IEC 14496-1. It can be a standalone system, or part of an application system complying with ISO/IEC 14496.

4.60 Time Base

The notion of a clock; it is equivalent to a counter that is periodically incremented.

4.61 Timing Model

A model that specifies the semantic meaning of timing information, how it is incorporated (explicitly or implicitly) in the coded representation of information, and how it can be recovered at the receiving terminal.

4.62 Time Stamp

An indication of a particular time instant relative to a time base.

5 Abbreviations and Symbols

AU	Access Unit
AV	Audio-visual
BIFS	Binary Format for Scene
CM	Composition Memory
CTS	Composition Time Stamp
CU	Composition Unit
DAI	DMIF Application Interface (see ISO/IEC 14496-6)
DB	Decoding Buffer
DTS	Decoding Time Stamp
ES	Elementary Stream
ESI	Elementary Stream Interface
ESID	Elementary Stream Identifier
FAP	Facial Animation Parameters
FAPU	FAP Units
FDP	Facial Definition Parameters
FIG	FAP Interpolation Graph
FIT	FAP Interpolation Table
FMC	FlexMux Channel
FMOD	The floating point modulo (remainder) operator which returns the remainder of x/y such that: $\text{Fmod}(x/y) = x - k*y, \text{ where } k \text{ is an integer,}$ $\text{sgn}(\text{fmod}(x/y)) = \text{sgn}(x), \text{ and}$ $\text{abs}(\text{fmod}(x/y)) < \text{abs}(y)$
IP	Intellectual Property
IPI	Intellectual Property Identification
IPMP	Intellectual Property Management and Protection
NCT	Node Coding Tables
NDT	Node Data Type
NINT	Nearest INTEger value
OCI	Object Content Information
OCR	Object Clock Reference
OD	Object Descriptor
ODID	Object Descriptor Identifier
OTB	Object Time Base
PLL	Phase Locked Loop
QoS	Quality of Service
SAOL	Structured Audio Orchestra Language
SASL	Structured Audio Score Language
SDL	Syntactic Description Language
SDM	Systems Decoder Model
SL	Synchronization Layer
SL-Packet	Synchronization Layer Packet
SPS	SL-Packetized Stream
STB	System Time Base
TTS	Text-To-Speech
URL	Universal Resource Locator
VOP	Video Object Plane
VRML	Virtual Reality Modeling Language

6 Conventions

For the purpose of unambiguously defining the syntax of the various bitstream components defined by the normative parts of ISO/IEC 14496 a *syntactic description language* is used. This language allows the specification of the mapping of the various parameters in a binary format as well as how they are placed in a serialized bitstream. The definition of the language is provided in clause 12.

7 Systems Decoder Model

7.1 Introduction

The purpose of the systems decoder model (SDM) is to provide an abstract view of the behavior of a terminal complying with ISO/IEC 14496. It may be used by the sender to predict how the receiving terminal will behave in terms of buffer management and synchronization when decoding data received in the form of elementary streams. The systems decoder model includes a timing model and a buffer model.

The systems decoder model specifies:

1. the interface for accessing demultiplexed data streams (DMIF Application Interface),
2. decoding buffers for coded data for each elementary stream,
3. the behavior of elementary stream decoders,
4. composition memory for decoded data from each decoder, and
5. the output behavior of composition memory towards the compositor.

These elements are depicted in Figure 2. Each elementary stream is attached to one single decoding buffer. More than one elementary stream may be connected to a single decoder (e.g., in a decoder of a scaleable audio-visual object).

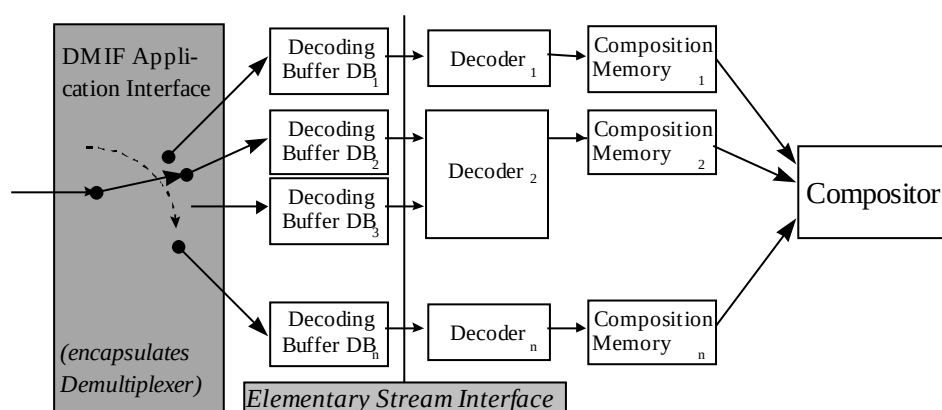


Figure 2 - Systems Decoder Model

7.2 Concepts of the Systems Decoder Model

This subclause defines the concepts necessary for the specification of the timing and buffering model. The sequence of definitions corresponds to a walk from the left to the right side of the SDM illustration in Figure 2.

7.2.1 DMIF Application Interface (DAI)

For the purposes of the systems decoder model, the DMIF Application Interface encapsulates the demultiplexer and provides access to streaming data that is consumed by the decoding buffers. The streaming data received

through the DAI consists of SL-packetized streams. The required properties of the DAI are described in 10.4. The DAI semantics are fully specified in ISO/IEC 14496-6.

7.2.2 SL-Packetized Stream (SPS)

An SL-packetized stream consists of a sequence of packets, according to the syntax and semantics specified in 10.2, that encapsulate a single elementary stream. The packets contain elementary stream data partitioned in access units as well as side information, e.g., for timing and access unit labeling. SPS data enter the decoding buffers.

7.2.3 Access Units (AU)

Elementary stream data is partitioned into access units. The delineation of an access unit is completely determined by the entity that generates the elementary stream (e.g., the compression layer). An access unit is the smallest data entity to which timing information can be attributed. Two access units from the same elementary stream shall never refer to the same decoding or composition time. Any further partitioning of the data in an elementary stream is not visible for the purposes of the Systems Decoder Model. Access units are conveyed by SL-packetized streams and are received by the decoding buffers. The decoders consume access units with the necessary side information (e.g., time stamps) from the decoding buffers.

NOTE — An ISO/IEC 14496-1 compliant terminal implementation is not required to process each incoming access unit as a whole. It is furthermore possible to split an access unit into several fragments for transmission as specified in clause 10. This allows the sending terminal to dispatch partial AUs immediately as they are generated during the encoding process. Such partial AUs may have significance for improved error resilience.

7.2.4 Decoding Buffer (DB)

The decoding buffer is a buffer at the input of an elementary stream decoder in the receiving terminal that receives and stores access units. The Systems Buffer Model enables the sending terminal to monitor the decoding buffer resources that are used during a presentation.

7.2.5 Elementary Streams (ES)

Streaming data received at the output of a decoding buffer, independent of its content, is considered as an elementary stream for the purpose of ISO/IEC 14496. The elementary streams are produced and consumed by the compression layer entities (encoders and decoders, respectively). ISO/IEC 14496 assumes that the integrity of an elementary stream is preserved from end to end, from the ESI of the sending terminal to the ESI of the receiving terminal.

7.2.6 Elementary Stream Interface (ESI)

The elementary stream interface models the exchange of elementary stream data and associated control information between the compression layer and the sync layer. At the receiving terminal the ESI is located at the output of the decoding buffer. The ESI is specified in 10.3.

7.2.7 Decoder

For the purposes of this model, the decoder extracts access units from the decoding buffer at precisely defined points in time and places composition units, the results of the decoding processes, in the composition memory. A decoder may be attached to several decoding buffers.

7.2.8 Composition Units (CU)

Decoders consume access units and produce composition units. An access unit corresponds to an integer number of composition units. Composition units reside in composition memory.

7.2.9 Composition Memory (CM)

The composition memory is a random access memory that contains composition units. The size of this memory is not normatively specified.

7.2.10 Compositor

The compositor takes composition units out of the composition memory and either consumes them (e.g. composes and presents them, in the case of audio-visual data) or skips them. The compositor is not specified in ISO/IEC 14496-1, as the details of this operation are not relevant within the context of the System Decoder Model. Subclause 7.3.5 defines which composition units are available to the compositor at any instant of time.

7.3 Timing Model Specification

The timing model relies on clock references and time stamps to synchronize audio-visual data conveyed by one or more elementary streams. The concept of a clock with its associated clock references is used to convey the notion of time to a receiving terminal. Time stamps are used to indicate the precise time instants at which the receiving terminal consumes the access units in the decoding buffers or may access the composition units resident in the composition memory. The time stamps are therefore associated with access units and composition units. The semantics of the timing model are defined in the subsequent clauses. The syntax for conveying timing information is specified in 10.2.

NOTE — This timing model is designed for rate-controlled (“push”) applications.

7.3.1 System Time Base (STB)

The System Time Base (STB) defines the terminal's notion of time. The resolution of the STB is implementation dependent. All actions of the terminal are scheduled according to this time base for the purpose of this timing model.

NOTE — This does not imply that all terminals compliant with ISO/IEC 14496 operate on one single STB.

7.3.2 Object Time Base (OTB)

The object time base (OTB) defines the notion of time for a given data stream. The resolution of this OTB can be selected as required by the application or as defined by a profile. All time stamps that the sending terminal inserts in a coded data stream refer to this time base. The OTB of a data stream is known at the receiving terminal either by means of object clock reference information inserted in the stream or by an indication that its time base is slaved to a time base conveyed with another stream, as specified in 10.2.3.

NOTE 1 — Elementary streams may be created for the sole purpose of conveying time base information.

NOTE 2 — The receiving terminal's System Time Base need not be locked to any of the available object time bases.

7.3.3 Object Clock Reference (OCR)

A special kind of time stamps, object clock references (OCR), are used to convey the OTB to the elementary stream decoder. The value of the OCR corresponds to the value of the OTB at the time the sending terminal generates the object clock reference time stamp. OCR time stamps are placed in the SL packet header as described in 10.2.4. The receiving terminal shall extract and evaluate the OCR when its first byte enters its decoding buffer.

7.3.4 Decoding Time Stamp (DTS)

Each access unit has an associated nominal decoding time, the time at which it must be available in the decoding buffer for decoding. The AU is not guaranteed to be available in the decoding buffer either before or after this time. Decoding is assumed to occur instantaneously when the instant of time indicated by the DTS is reached.

This point in time can be implicitly specified if the (constant) temporal distance between successive access units is indicated in the setup of the elementary stream (see 10.2.3). Otherwise a decoding time stamp (DTS) whose syntax is defined in 10.2.4 conveys this point in time.

A decoding time stamp shall only be conveyed for an access unit that carries a composition time stamp as well, and only if the DTS and CTS values are different. Presence of both time stamps in an AU may indicate a reversal between coding order and composition order.

7.3.5 Composition Time Stamp (CTS)

Each composition unit has an associated nominal composition time, the time at which it must be available in the composition memory for composition. The CU is not guaranteed to be available in the composition memory *for composition* before this time. Since the SDM assumes an instantaneous decoding process, the CU is available to the *decoder*, at that instant in time corresponding to the DTS of the corresponding AU, for further use (e.g. in prediction processes).

This instant in time is implicitly known, if the (constant) temporal distance between successive composition units is indicated in the setup of the elementary stream. Otherwise a composition time stamp (CTS) whose syntax is defined in 10.2.4 conveys this instant in time.

The current CU is instantaneously accessible by the compositor anytime between its composition time and the composition time of the subsequent CU. If a subsequent CU does not exist, the current CU becomes unavailable at the end of the lifetime of its elementary stream (i.e., when its elementary stream descriptor is removed).

7.3.6 Occurrence and Precision of Timing Information in Elementary Streams

The frequency at which DTS, CTS and OCR values are to be inserted in the bitstream as well as the precision, jitter and drift are application and profile dependent. Some usage considerations can be found in 10.2.7.

7.3.7 Time Stamps for Dependent Elementary Streams

An audio-visual object may refer to multiple elementary streams that constitute a scaleable content representation (see 8.7.1.5). Such a set of elementary streams shall adhere to a single object time base. Temporally co-located access units for such elementary streams are then identified by identical DTS or CTS values.

EXAMPLE

The example in Figure 3 illustrates the arrival of two access units at the Systems Decoder. Due to the constant delay assumption of the model (see 7.4.2 below), the arrival times correspond to the instants in time when the sending terminal has sent the respective AUs. The sending terminal must select this instant in time so that the Decoding Buffer at the receiving terminal never overflows or underflows. At the receiving terminal, an AU is instantaneously decoded, at that instant in time corresponding to its DTS, and the resulting CU(s) are placed in the composition memory and remain there until the subsequent CU(s) arrive or the associated object descriptor is removed.

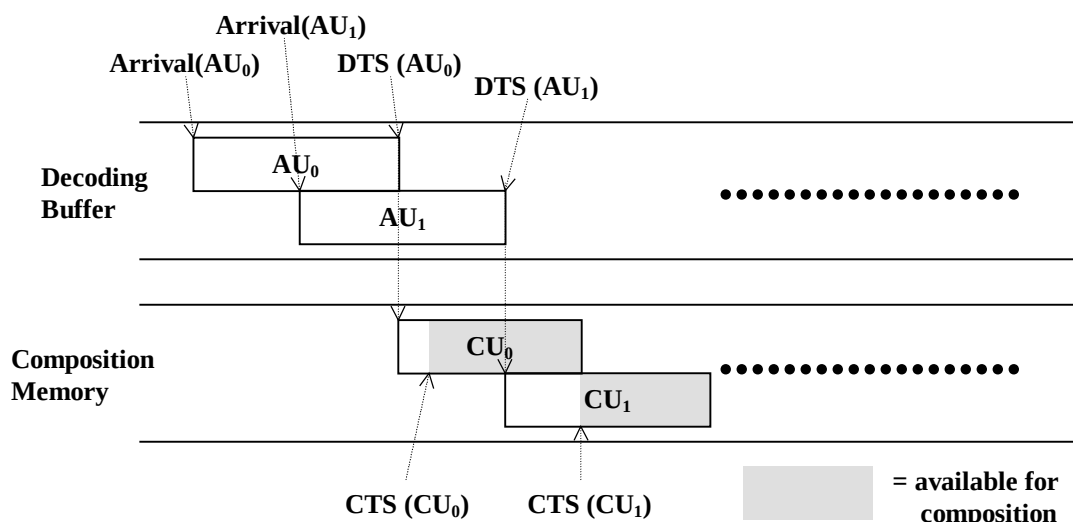


Figure 3 - Composition unit availability

7.4 Buffer Model Specification

7.4.1 Elementary Decoder Model

Figure 4 indicates one branch of the Systems Decoder Model (Figure 2). This simplified model is used to specify the buffer model. It treats each elementary stream separately and therefore, associates a composition memory with only one decoder. The legend following Figure 4 elaborates on the symbols used in this figure.

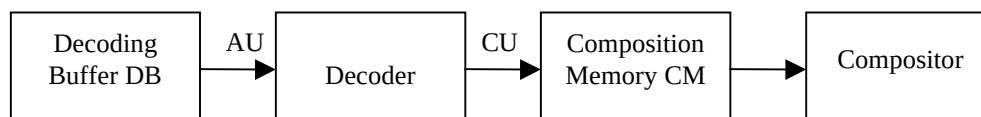


Figure 4 - Flow diagram for the Systems Decoder Model

Legend:

- DB Decoding buffer for the elementary stream.
- CM Composition memory for the elementary stream.
- AU The current access unit input to the Decoder.
- CU The current composition unit input to the composition memory. CU results from decoding AU. There may be several composition units resulting from decoding one access unit.

7.4.2 Assumptions

7.4.2.1 Constant end-to-end delay

Data transmitted in real time have a timing model in which the end-to-end delay from the encoder input at the sending terminal, to the decoder output at the receiving terminal, is constant. This delay is equal to the sum of the delay due to the encoding process, subsequent buffering, multiplexing at the sending terminal, the delay due to the delivery layers and the delay due to the demultiplexing, decoder buffering and decoding processes at the receiving terminal.

Note that the receiving terminal is free to add a temporal offset (delay) to the absolute values of all time stamps if it can cope with the additional buffering needed. However, the temporal difference between two time stamps (that determines the temporal distance between the associated AUs or CUs) has to be preserved for real-time performance.

7.4.2.2 Demultiplexer

The end-to-end delay between multiplexer output, at the sending terminal, and demultiplexer input, at the receiving terminal, is constant.

7.4.2.3 Decoding Buffer

The needed decoding buffer size is known by the sending terminal and conveyed to the receiving terminal as specified in 8.6.5.

The size of the decoding buffer is measured in bytes.

The decoding buffer is filled at the rate given by the maximum bit rate for this elementary stream (if this information is conveyed by the sending terminal), and with a zero rate otherwise. The maximum bit rate is conveyed by the

sending terminal as a part of the decoder configuration information during the set up phase for each elementary stream (see 8.6.5).

Information is received from the DAI in the form of SL packets. The SL packet headers are removed at the input to the decoding buffers.

7.4.2.4 Decoder

The decoding processes are assumed to be instantaneous for the purposes of the Systems Decoder Model.

7.4.2.5 Composition Memory

The mapping of an AU to one or more CUs (by the decoder) is known implicitly at both the sending and the receiving terminals.

7.4.2.6 Compositor

The composition processes are assumed to be instantaneous for the purposes of the Systems Decoder Model.

7.4.3 Managing Buffers: A Walkthrough

In this example, we assume that the model is used in a “push” scenario. In applications where non-real time content is to be delivered, flow control by suitable signaling may be established to request access units at the time they are needed at the receiving terminal. The mechanisms for doing so are application-dependent, and are not specified in ISO/IEC 14496.

The behaviors of the various elements in the SDM are modeled as follows:

- The sending terminal signals the required decoding buffer resources to the receiving terminal before starting the delivery. This is done as specified in 8.6.5 either explicitly, by requesting the decoding buffer sizes for individual elementary streams, or implicitly, by indicating a profile (see clause 13). The decoding buffer size is measured in bytes.
- The sending terminal models the behavior of the decoding buffers by making the following assumptions :
- Each decoding buffer is filled at the maximum bitrate specified for its associated elementary stream if this information is available.
- At the instant of time corresponding to its DTS, an AU is instantaneously decoded and removed from the decoding buffer.
- At the instant of time corresponding to its DTS, a known amount of CUs corresponding to the just decoded AU are put in the composition memory.

The current CU is available to the compositor between instants of time corresponding to the CTS of the current CU and the CTS of the subsequent CU. If a subsequent CU does not exist, the current CU becomes unavailable at the end of lifetime of its data stream.

Using these assumptions on the buffer model, the sending terminal may freely use the space in the decoding buffers. For example, it may deliver data for several AUs of a stream, for non real time usage, to the receiving terminal, and pre-store them in the DB long before they have to be decoded (assuming sufficient space is available). Subsequently, the full delivery bandwidth may be used to transfer data of a real time stream just in time. The composition memory may be used, for example, as a reordering buffer. In the case of visual decoding, it may contain the decoded P-frames needed by a video decoder for the decoding of intermediate B-frames, before the arrival of the CTS of the latest P-frame.

8 Object Description Framework

8.1 Introduction

The scene description (specified in clause 9) and the elementary streams that convey streaming data are the basic building blocks of the architecture of ISO/IEC 14496-1. Elementary streams carry data for audio or visual objects as well as for the scene description itself. The object description framework provides the link between elementary streams and the scene description. The scene description declares the spatio-temporal relationship of audio-visual objects, while the object description framework specifies the elementary stream resources that provide the time-varying data for the scene. This indirection facilitates independent changes to the scene structure, the properties of the elementary streams (e.g. its encoding) and their delivery.

The object description framework consists of a set of descriptors that allows to identify, describe and properly associate elementary streams to each other and to audio-visual objects used in the scene description. Numeric identifiers, called ObjectDescriptorIDs, associate object descriptors to appropriate nodes in the scene description. Object descriptors are themselves conveyed in elementary streams to allow time stamped changes to the available set of object descriptors to be made.

Each object descriptor is itself a collection of descriptors that describe one or more elementary streams that are associated to a single node and that usually relate to a single audio or visual object. This allows to indicate a scaleable content representation as well as multiple alternative streams that convey the same content, e.g., in multiple qualities or different languages.

An elementary stream descriptor within an object descriptor identifies a single elementary stream with a numeric identifier, called ES_ID. Each elementary stream descriptor contains the information necessary to initiate and configure the decoding process for the elementary stream, as well as intellectual property identification. Optionally, additional information may be associated to a single elementary stream, most notably quality of service requirements for its transmission or a language indication. Both, object descriptors and elementary stream descriptors may use URLs to point to remote object descriptors or a remote elementary stream source, respectively.

The object description framework provides the hooks to implement intellectual property management and protection (IPMP) systems. IPMP information is conveyed both through IPMP descriptors as part of the object descriptor stream and through IPMP streams that carry time variant IPMP information. The structure of IPMP descriptors and IPMP streams is specified in this clause while their internal syntax and semantics and, hence, the operation of the IPMP system is outside the scope of ISO/IEC 14496.

Object content information allows the association of metadata with a whole presentation or with individual object descriptors or with elementary stream descriptors. A set of OCI descriptors is defined that either form an integral part of an object descriptor or elementary stream descriptor or are conveyed by means of a proper OCI stream that allows the conveyance of time variant object content information.

Access to ISO/IEC 14496 content is gained through an initial object descriptor that needs to be made available through means not defined in ISO/IEC 14496. The initial object descriptor in the simplest case points to the scene description stream and the corresponding object descriptor stream. The access scenario is outlined in 8.7.3.

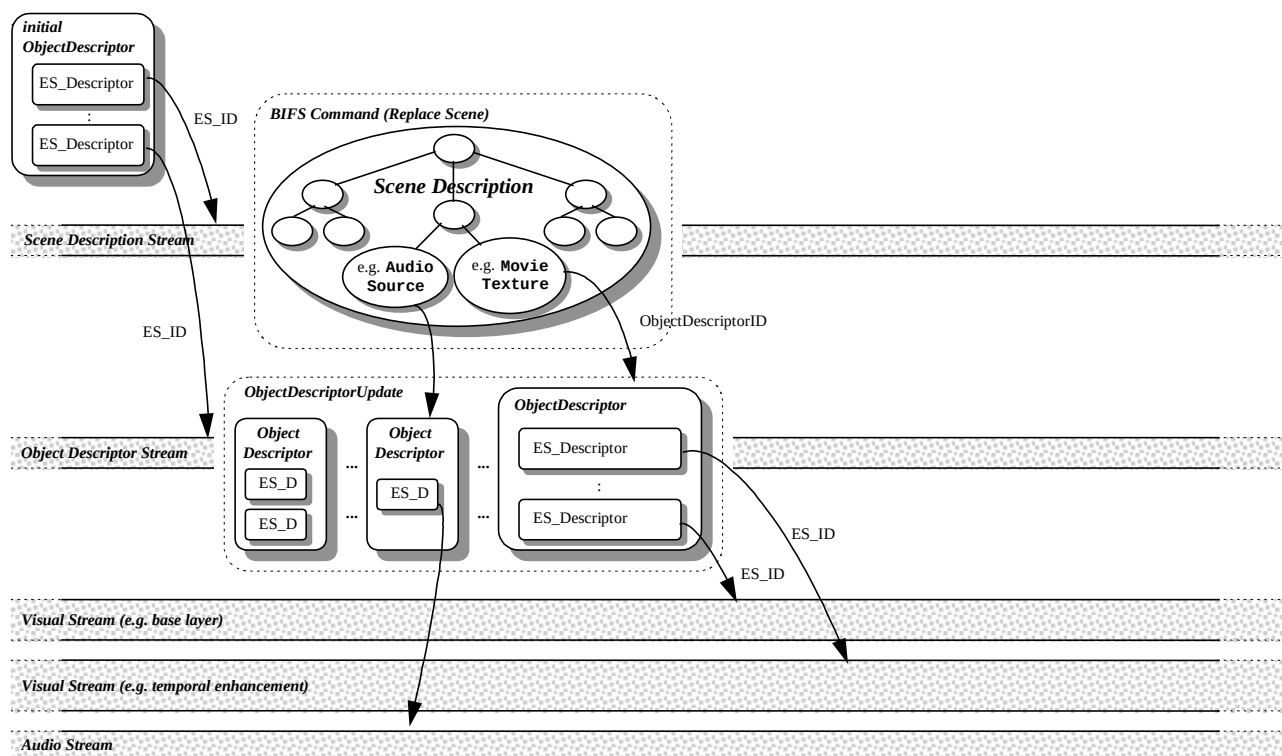


Figure 5 - Object descriptors linking scene description to elementary streams

The remainder of this clause is structured in the following way:

- Subclause 8.2 specifies the data structures on which the object descriptor framework is based.
- Subclause 8.3 specifies the concepts of the IPMP elements in the object description framework.
- Subclause 8.4 specifies the object content information elements in the object description framework.
- Subclause 8.5 specifies the object descriptor stream and the syntax and semantics of the command set that allows the update or removal of object descriptor components.
- Subclause 8.6 specifies the syntax and semantics of the object descriptor and its component descriptors.
- Subclause 8.7 specifies rules for object descriptor usage as well as the procedure to access content through object descriptors.
- Subclause 8.8 specifies the usage of the IPMP system interface.

8.2 Common data structures

8.2.1 Overview

The commands and descriptors defined in this subclause constitute self-describing classes, identified by unique class tags. Each class encodes explicitly its size in bytes. This facilitates future compatible extensions of the commands and descriptors. They may be expanded with additional syntax elements that are ignored by an OD decoder that expects an earlier revision of a class. In addition, anywhere in a syntax where a set of tagged classes is expected it is permissible to intersperse expandable classes with unknown class tag values. These classes shall be skipped, using the encoded size information.

The remainder of this clause defines the syntax and semantics of the command and descriptor classes. Some commands and descriptors contain themselves a set of component descriptors. They are said to *aggregate a set of component descriptors*.

8.2.2 BaseDescriptor

8.2.2.1 Syntax

```
abstract aligned(8) expandable( $2^{28}-1$ ) class BaseDescriptor : bit(8) tag=0 {
    // empty. To be filled by classes extending this class.
}
```

8.2.2.2 Semantics

This class is an abstract base class that is extended by the descriptor classes specified in 8.6. Each descriptor constitutes a self-describing class, identified by a unique class tag. This abstract base class establishes a common name space for the class tags of these descriptors. The values of the class tags are defined in Table 1. As an expandable class the size of each class instance in bytes is encoded and accessible through the instance variable `sizeofInstance` (see 12.3.3).

A class that allows the aggregation of classes of type `BaseDescriptor` may actually aggregate any of the classes that extend `BaseDescriptor`.

NOTE — User private descriptors may have an internal structure, for example to identify the country or manufacturer that uses a specific descriptor. The tags and semantics for such user private descriptors may be managed by a registration authority if required.

Table 1 - List of Class Tags for Descriptors

Tag value	Tag name
0x00	Forbidden
0x01	ObjectDescrTag
0x02	InitialObjectDescrTag
0x03	ES_DescrTag
0x04	DecoderConfigDescrTag
0x05	DecSpecificInfoTag
0x06	SLConfigDescrTag
0x07	ContentIdentDescrTag
0x08	SupplContentIdentDescrTag
0x09	IPI_DescrPointerTag
0x0A	IPMP_DescrPointerTag
0x0B	IPMP_DescrTag
0x0C	QoS_DescrTag
0x0D	RegistrationDescrTag
0x0E-0x3F	Reserved for ISO use (descriptors)
0x40	ContentClassificationDescrTag
0x41	KeyWordDescrTag
0x42	RatingDescrTag
0x43	LanguageDescrTag
0x44	ShortTextualDescrTag
0x45	ExpandedTextualDescrTag
0x46	ContentCreatorNameDescrTag
0x47	ContentCreationDateDescrTag
0x48	OCICreatorNameDescrTag
0x49	OCICreationDateDescrTag
0x4A-0x5F	Reserved for ISO use (OCI extensions)
0x60-0xBF	Reserved for ISO use
0xC0-0xFE	User private
0xFF	Forbidden

The following additional symbolic names are introduced:

```
ExtDescrTagStartRange = 0x80
ExtDescrTagEndRange = 0xFE
OCIDescrTagStartRange = 0x40
OCIDescrTagEndRange = 0x5F
```

8.2.3 BaseCommand

8.2.3.1 Syntax

```
abstract aligned(8) expandable(228-1) class BaseCommand : bit(8) tag=0 {
    // empty. To be filled by classes extending this class.
}
```

8.2.3.2 Semantics

This class is an abstract base class that is extended by the command classes specified in 8.5.5. Each command constitutes a self-describing class, identified by a unique class tag. This abstract base class establishes a common name space for the class tags of these commands. The values of the class tags are defined in Table 2. As an expandable class the size of each class instance in bytes is encoded and accessible through the instance variable `sizeofInstance` (see 12.3.3).

A class that allows the aggregation of classes of type `BaseCommand` may actually aggregate any of the classes that extend `BaseCommand`.

NOTE — User private commands may have an internal structure, for example to identify the country or manufacturer that uses a specific command. The tags and semantics for such user private command may be managed by a registration authority if required.

Table 2 - List of Class Tags for Commands

Tag value	Tag name
0x00	forbidden
0x01	ObjectDescrUpdateTag
0x02	ObjectDescrRemoveTag
0x03	ES_DescrUpdateTag
0x04	ES_DescrRemoveTag
0x05	IPMP_DescrUpdateTag
0x06	IPMP_DescrRemoveTag
0x07-0xBF	Reserved for ISO (command tags)
0xC0-0xFE	User private
0xFF	forbidden

8.3 Intellectual Property Management and Protection (IPMP)

8.3.1 Overview

The intellectual property management and protection (IPMP) framework for ISO/IEC 14496 content consists of a normative interface that permits an ISO/IEC 14496 terminal to host one or more IPMP Systems. An IPMP System is a non-normative component that provides intellectual property management and protection functions for the terminal.

The IPMP interface consists of IPMP elementary streams and IPMP descriptors. The normative structure of IPMP elementary streams is specified in this subclause. IPMP descriptors are carried as part of an object descriptor stream and are specified in 8.6.13. The IPMP interface allows applications (or derivative application standards) to build specialized IPMP Systems. Alternatively, an application may choose not to use an IPMP System, thereby offering no management and protection features. The IPMP System uses the information carried by the IPMP elementary streams and descriptors to make protected ISO/IEC 14496 content available to the terminal. The detailed semantics and decoding process of the IPMP System are not in the scope of ISO/IEC 14496. The usage of the IPMP System Interface, however, is explained in 8.8.

8.3.2 IPMP Streams

8.3.2.1 Structure of the IPMP Stream

The IPMP stream is an elementary stream that passes time-varying information to one or more IPMP Systems. This is accomplished by periodically sending a sequence of IPMP messages along with the content at a period determined by the IPMP System(s).

8.3.2.2 Access Unit Definition

An IPMP access unit consists of one or more IPMP messages, as defined in 8.3.2.5. All IPMP messages that are to be processed at the same instant in time shall constitute a single access unit. Access units in IPMP streams shall be labeled and time-stamped by suitable means. This shall be done via the related flags and the composition time stamps, respectively, in the SL packet header (see 10.2.4). The composition time indicates the point in time at which an IPMP access unit becomes valid, i.e., when the embedded IPMP messages shall be evaluated. Decoding and composition time for an IPMP access unit shall always have the same value.

An access unit does not necessarily convey or update the complete set of IPMP messages that are currently required. In that case it just modifies the persistent state of the IPMP system. However, if an access unit conveys the complete set of IPMP messages required at a given point in time it shall set the `randomAccessPointFlag` in the SL packet header to '1' for this access unit. Otherwise, the `randomAccessPointFlag` shall be set to '0'.

NOTE — An SL packet with `randomAccessPointFlag`=1 but with no IPMP messages in it indicates that at the current time instant no IPMP messages are required for operation.

8.3.2.3 Time Base for IPMP Streams

The time base associated to an IPMP stream shall be indicated by suitable means. This shall be done by means of object clock reference time stamps in the SL packet headers (see 10.2.4) for this stream or by indicating the elementary stream from which this IPMP stream inherits the time base (see 10.2.3). All time stamps in the SL-packetized IPMP stream refer to this time base.

8.3.2.4 IPMP Decoder Configuration

8.3.2.4.1 Syntax

```
class IPMPDecoderConfiguration extends DecoderSpecificInfo : bit(8) tag=DecSpecificInfoTag {
    // IPMP system specific configuration information
}
```

8.3.2.4.2 Semantics

An IPMP system may require information to initialize its operation. This information shall be conveyed by extending the `decoderSpecificInfo` class as specified in 8.6.6. If utilized, `IPMPDecoderConfiguration` shall be conveyed in the `ES_Descriptor` declaring the IPMP stream.

8.3.2.5 IPMP message syntax and semantics

8.3.2.5.1 Syntax

```
class IPMP_Message() extends ExpandableBaseClass
{
    bit(16) IPMPS_Type;
    if (IPMPS_Type == 0) {
        bit(8) URLString[sizeofInstance-2];
    } else {
        bit(8) IPMP_data[sizeofInstance-2];
    }
}
```

8.3.2.5.2 Semantics

The `IPMP_Message` conveys control information for an IPMP System.

IPMP_Type - the type of the IPMP System. A zero value does not correspond to an IPMP System, but indicates the presence of a URL. A Registration Authority as designated by the ISO shall assign valid values for this field.

URLString[] - contains a UTF-8 [3] encoded URL that shall point to the location of a remote IPMP_Message whose IPMP_data shall be used in place of locally provided data.

IPMP_data - opaque data to control the IPMP System.

8.4 Object Content Information (OCI)

8.4.1 Overview

Audio-visual objects that are associated with elementary stream data through an object descriptor may have additional object content information attached to them. For this purpose, a set of OCI descriptors is defined in 8.6.17. OCI descriptors may directly be included as part of an ObjectDescriptor or ES_Descriptor as defined in 8.6.1.

In order to accommodate time variant OCI that is separable from the object descriptor stream, OCI descriptors may as well be conveyed in an OCI stream. An OCI stream is referred to through an ES_Descriptor, with the streamType field set to OCI_Stream. How OCI streams may be aggregated to object descriptors is defined in 8.7.1.3. The structure of the OCI stream is defined in this subclause.

8.4.2 OCI Streams

8.4.2.1 Structure of the OCI Stream

The OCI stream is an elementary stream that conveys time-varying object content information, termed OCI events. Each OCI event consists of a number of OCI descriptors.

8.4.2.2 Access Unit Definition

An OCI access unit consists of one or more OCI_Events, as described in 8.4.2.5. Access units in OCI elementary streams shall be labelled and time stamped by suitable means. This shall be done by means of the related flags and the composition time stamp, respectively, in the SL packet header (see 10.2.4). The composition time indicates the point in time when an OCI access unit becomes valid, i.e., when the embedded OCI events shall be added to the list of events. Decoding and composition time for an OCI access unit shall always have the same value.

An access unit may or may not convey or update the complete set of OCI events that are currently valid. In the latter case, it just modifies the persistent state of the OCI decoder. However, if an access unit conveys the complete set of OCI events valid at a given point in time it shall set the randomAccessPointFlag in the SL packet header to '1' for this access unit. Otherwise, the randomAccessPointFlag shall be set to '0'.

NOTE — An SL packet with randomAccessPointFlag=1 but with no OCI events in it indicates that at the current time instant no valid OCI events exist.

8.4.2.3 Time Base for OCI Streams

The time base associated with an OCI stream shall be indicated by suitable means. This shall be done by the use of object clock reference time stamps in the SL packet headers (see 10.2.4) for this stream or by indicating the elementary stream from which this OCI stream inherits the time base (see 10.2.3). All time stamps in the SL-packetized OCI stream refer to this time base.

8.4.2.4 OCI Decoder Configuration

8.4.2.4.1 Syntax

```
class OCIDecoderConfiguration extends DecoderSpecificInfo : bit(8) tag=DecSpecificInfoTag {  
    const bit(8) versionLabel = 0x01;  
}
```

8.4.2.4.2 Semantics

This information is needed to initialize operation of the OCI decoder. It shall be conveyed by extending the `decoderSpecificInfo` class as specified in 8.6.6. `OCIDecoderConfiguration` shall be conveyed in the `ES_Descriptor` declaring the OCI stream.

`versionLabel` – indicates the version of OCI specification used on the corresponding OCI data stream. Only the value 0x01 is allowed; all the other values are reserved.

8.4.2.5 OCI_Events syntax and semantics

8.4.2.5.1 Syntax

```
class OCI_Event extends ExpandableBaseClass {
    bit(15) eventID;
    bit(1) absoluteTimeFlag;
    bit(32) startingTime;
    bit(32) duration;
    OCI_Descriptor OCI_Descr[1 .. 255];
}
```

8.4.2.5.2 Semantics

`eventID` – contains the identification number of the described event that is unique within the scope of this OCI stream.

`absoluteTimeFlag` – indicates the time base for `startingTime` as described below.

`startingTime` – indicates the starting time of the event in hours, minutes, seconds and hundredth of seconds. The format is 8 digits, the first 6 digits expressing hours, minutes and seconds with 4 bits each in binary coded decimal and the last two expressing hundredth of seconds in hexadecimal using 8 bits.

EXAMPLE — 02:36:45:89 is coded as “0x023645” concatenated with “0b0101.1001” (89 in binary), resulting to “0x02364559”.

If `absoluteTimeFlag` is set to zero, `startingTime` is relative to the object time base of the corresponding object. In that case it is the responsibility of the application to ensure that this object time base is conveyed such that `startingTime` can be identified unambiguously (see 10.2.7). If `absoluteTimeFlag` is set to one, `startingTime` is expressed as an absolute value, referring to wall clock time.

`duration` – contains the duration of the corresponding object in hours, minutes, seconds and hundredth of seconds. The format is 8 digits, the first 6 digits expressing hours, minutes and seconds with 4 bits each in binary coded decimal and the last two expressing hundredth of seconds in hexadecimal using 8 bits.

`OCI_Descr[]` – an array of one up to 255 `OCI_Descriptor` classes as specified in 8.6.17.2.

8.5 Object Descriptor Stream

8.5.1 Structure of the Object Descriptor Stream

Similar to the scene description, object descriptors are transported in a dedicated elementary stream, termed object descriptor stream. Within such a stream, it is possible to dynamically convey, update and remove complete object descriptors, or their component descriptors, the `ES_Descriptors`, and `IPMP` descriptors. The update mechanism allows, for example, to advertise new elementary streams for an audio-visual object as they become available, or to remove references to streams that are no longer available. Updates are time stamped to indicate the instant in time they take effect.

This subclause specifies the structure of the object descriptor elementary stream including the syntax and semantics of its constituent elements, the object descriptor commands (OD commands).

8.5.2 Access Unit Definition

An OD access unit consists of one or more OD commands, as described in 8.5.5. All OD commands that are to be processed at the same instant in time shall constitute a single access unit. Access units in object descriptor elementary streams shall be labelled and time stamped by suitable means. This shall be done by means of the

related flags and the composition time stamp, respectively, in the SL packet header (see 10.2.4). The composition time indicates the point in time when an OD access unit becomes valid, i.e., when the embedded OD commands shall be executed. Decoding and composition time for an OD access unit shall always have the same value.

An access unit may not convey or update the complete set of object descriptors that are currently required. In that case it just modifies the persistent state of the object descriptor decoder. However, if an access unit conveys the complete set of object descriptors required at a given point in time it shall set the `randomAccessPointFlag` in the SL packet header to '1' for this access unit. Otherwise, the `randomAccessPointFlag` shall be set to '0'.

NOTE — An SL packet with `randomAccessPointFlag=1` but with no OD commands in it indicates that at the current time instant no valid object descriptors exist.

8.5.3 Time Base for Object Descriptor Streams

The time base associated to an object descriptor stream shall be indicated by suitable means. This shall be done by means of object clock reference time stamps in the SL packet headers (see 10.2.4) for this stream or by indicating the elementary stream from which this object descriptor stream inherits the time base (see 10.2.3). All time stamps in the SL-packetized object descriptor stream refer to this time base.

8.5.4 OD Decoder Configuration

The object descriptor decoder does not require additional configuration information.

8.5.5 OD Command Syntax and Semantics

8.5.5.1 Overview

Object descriptors and their components as defined in 8.6 shall always be conveyed as part of one of the OD commands specified in this subclause. The commands describe the action to be taken on the components conveyed with the command, specifically 'update' or 'remove'. Each command affects one or more object descriptors, ES_Descriptors or IPMP descriptors.

8.5.5.2 ObjectDescriptorUpdate

8.5.5.2.1 Syntax

```
class ObjectDescriptorUpdate extends BaseCommand : bit(8) tag=ObjectDescrUpdateTag {
    ObjectDescriptor OD[1 .. 255];
}
```

8.5.5.2.2 Semantics

The `ObjectDescriptorUpdate` class conveys a list of new or updated `ObjectDescriptors`. The components of an already existing `ObjectDescriptor` shall not be changed by an update, but an `ObjectDescriptorUpdate` may remove or add `ES_Descriptors` as components of the related object descriptor.

Removal of an `ES_Descriptor` within an `ObjectDescriptor` conveyed by this command is accomplished by omitting it from the array of `ES_Descriptors` aggregated to the `ObjectDescriptor`. Addition of an `ES_Descriptor` within an `ObjectDescriptor` conveyed by this command is accomplished by adding it to the array of `ES_Descriptors` aggregated to the `ObjectDescriptor`.

To update the characteristics of an elementary stream, it is required that its original `ES_Descriptor` be removed and the changed `ES_Descriptor` be conveyed.

`OD[]` – an array of `ObjectDescriptors` as defined in 8.6.2. The array shall have any number of one up to 255 elements.

8.5.5.3 ObjectDescriptorRemove

8.5.5.3.1 Syntax

```
class ObjectDescriptorRemove extends BaseCommand : bit(8) tag=ObjectDescrRemoveTag {
    bit(10) objectDescriptorId[(sizeofInstance*8)/10];
}
```

8.5.5.3.2 Semantics

The `ObjectDescriptorRemove` class renders unavailable a set of object descriptors. The BIFS nodes associated to these object descriptors shall have no reference any more to the elementary streams that have been listed in the removed object descriptors. An `objectDescriptorID` that does not refer to a valid `ObjectDescriptor` is ignored.

NOTE — It is possible that a scene description node references an `OD_ID` which does not currently have an associated OD.

`ObjectDescriptorId[]` – an array of `ObjectDescriptorIDs` that indicates the object descriptors that are removed.

8.5.5.4 ES_DescriptorUpdate

8.5.5.4.1 Syntax

```
class ES_DescriptorUpdate extends BaseCommand : bit(8) tag=ES_DescrUpdateTag {
    bit(10) objectDescriptorId;
    ES_Descriptor ESD[1 .. 30];
}
```

8.5.5.4.2 Semantics

The `ES_DescriptorUpdate` class adds or updates references to elementary streams within the object descriptor labeled `objectDescriptorID`. Values of syntax elements of an updated `ES_Descriptor` shall remain unchanged.

To update the characteristics of an elementary stream, it is required that its original `ES_Descriptor` be removed and the changed `ES_Descriptor` be conveyed.

An elementary stream identified with a given `ES_ID` may be attached to more than one object descriptor. All corresponding `ES_Descriptors` referring to this `ES_ID` that are conveyed through either `ES_DescriptorUpdate` or `ObjectDescriptorUpdate` commands shall have identical content.

`objectDescriptorID` - identifies the `ObjectDescriptor` for which `ES_Descriptors` are updated. If the `objectDescriptorID` does not refer to any valid object descriptor, then this command is ignored.

`ESD[]` – an array of `ES_Descriptors` as defined in 8.6.4. The array shall have any number of one up to 30 elements.

8.5.5.5 ES_DescriptorRemove

8.5.5.5.1 Syntax

```
class ES_DescriptorRemove extends BaseCommand : bit(8) tag=ES_DescrRemoveTag {
    bit(10) objectDescriptorId;
    aligned (8) bit(16) ES_ID[1..30];
}
```

8.5.5.5.2 Semantics

The `ES_DescriptorRemove` class removes the reference to an elementary stream from an `ObjectDescriptor` and renders this stream unavailable for nodes referencing this `ObjectDescriptor`.

`objectDescriptorID` - identifies the `ObjectDescriptor` from which `ES_Descriptors` are removed. If the `objectDescriptorID` does not refer to a valid object descriptor in the same scope, then this command is ignored.

`ES_ID[]` – an array of `streamCount` `ES_IDs` that labels the `ES_Descriptors` to be removed from `objectDescriptorID`. If any of the `ES_IDs` do not refer to an `ES_Descriptor` currently referenced by the OD, then those `ES_IDs` are ignored.

8.5.5.6 IPMP_DescriptorUpdate

8.5.5.6.1 Syntax

```
class IPMP_DescriptorUpdate extends BaseCommand : bit(8) tag=IPMP_DescrUpdateTag {
    IPMP_Descriptor ipmpDescr[1..255];
}
```

8.5.5.6.2 Semantics

The IPMP_DescriptorUpdate class conveys a list of new or updated IPMP_Descriptors. An IPMP_Descriptor identified by an IPMP_DescriptorID that already exists shall be replaced by the new descriptor.

IPMP_Descriptors remain valid until they are replaced by another IPMP_DescriptorUpdate command or removed.

ipmpDescr[] – an array of IPMP_Descriptor as specified in 8.6.13.

8.5.5.7 IPMP_DescriptorRemove

8.5.5.7.1 Syntax

```
class IPMP_DescriptorRemove extends BaseCommand : bit(8) tag=IPMP_DescrRemoveTag {
    bit(8) IPMP_DescriptorID[1..255];
}
```

8.5.5.7.2 Semantics

The IPMP_DescriptorRemove class conveys a list of IPMP_DescriptorsIDs that identify the IPMP_Descriptors that shall be removed.

IPMP_DescriptorID[] – is a list of IPMP_DescriptorIDs.

8.6 Object Descriptor Components

8.6.1 Overview

Object descriptors contain various additional descriptors as their components, in order to describe individual elementary streams and their properties. They shall always be conveyed as part of one of the OD commands specified in the previous subclause. This subclause defines the syntax and semantics of object descriptors and their component descriptors.

8.6.2 ObjectDescriptor

8.6.2.1 Syntax

```
class ObjectDescriptor extends BaseDescriptor : bit(8) tag=ObjectDescrTag {
    bit(10) ObjectDescriptorID;
    bit(1) URL_Flag;
    const bit(5) reserved=0b1111.1;
    if (URL_Flag) {
        bit(8) URLlength;
        bit(8) URLstring[URLlength];
    } else {
        ES_Descriptor esDescr[1 .. 30];
        OCI_Descriptor ociDescr[0 .. 255];
        IPMP_DescriptorPointer ipmpDescrPtr[0 .. 255];
    }
    ExtensionDescriptor extDescr[0 .. 255];
}
```

8.6.2.2 Semantics

The `ObjectDescriptor` consists of three different parts.

The first part uniquely labels the object descriptor within its name scope (see 8.7.2.4) by means of an `objectDescriptorId`. Nodes in the scene description use `objectDescriptorId` to refer to the related object descriptor. An optional `URLstring` indicates that the actual object descriptor resides at a remote location.

The second part consists of a list of `ES_Descriptors`, each providing parameters for a single elementary as well as an optional set of object content information descriptors and pointers to IPMP descriptors for the contents for elementary stream content described in this object descriptor.

The third part is a set of optional descriptors that support the inclusion of future extensions as well as the transport of private data in a backward compatible way.

`objectDescriptorId` – This syntax element uniquely identifies the `ObjectDescriptor` within its name scope. The value 0 is forbidden and the value 1023 is reserved.

`URL_Flag` – a flag that indicates the presence of a `URLstring`.

`URLlength` – the length of the subsequent `URLstring` in bytes.

`URLstring[]` – A string with a UTF-8 [3] encoded URL that shall point to another `ObjectDescriptor`. Only the content of this object descriptor shall be returned by the delivery entity upon access to this URL. Within the current name scope, the new object descriptor shall be referenced by the `objectDescriptorId` of the object descriptor carrying the `URLstring`. On name scopes see 8.7.2.4. Permissible URLs may be constrained by profile and levels as well as by specific delivery layers.

`esDescr[]` – an array of `ES_Descriptors` as defined in 8.6.4. The array shall have any number of one up to 30 elements.

`ociDescr[]` – an array of `OCI_Descriptors`, as defined in 8.6.17.2, that relates to the audio-visual object(s) described by this object descriptor. The array shall have any number of zero up to 255 elements.

`ipmpDescrPtr[]` – an array of `IPMP_DescriptorPointer`, as defined in 8.6.12, that points to the `IPMP_Descriptors` related to the elementary stream(s) described by this object descriptor. The array shall have any number of zero up to 255 elements.

`extDescr[]` – an array of `ExtensionDescriptors` as defined in 8.6.15. The array shall have any number of zero up to 255 elements.

8.6.3 InitialObjectDescriptor

8.6.3.1 Syntax

```
class InitialObjectDescriptor extends BaseDescriptor : bit(8) tag=InitialObjectDescrTag {
    bit(10) ObjectDescriptorID;
    bit(1) URL_Flag;
    bit(1) includeInlineProfileLevelFlag;
    const bit(4) reserved=0b1111;
    if (URL_Flag) {
        bit(8) URLlength;
        bit(8) URLstring[URLlength];
    } else {
        bit(8) ODProfileLevelIndication;
        bit(8) sceneProfileLevelIndication;
        bit(8) audioProfileLevelIndication;
        bit(8) visualProfileLevelIndication;
        bit(8) graphicsProfileLevelIndication;
        ES_Descriptor ESD[1 .. 30];
        OCI_Descriptor ociDescr[0 .. 255];
        IPMP_DescriptorPointer ipmpDescrPtr[0 .. 255];
    }
    ExtensionDescriptor extDescr[0 .. 255];
}
```

8.6.3.2 Semantics

The `InitialObjectDescriptor` is a variation of the `ObjectDescriptor` specified in the previous subclause that allows to signal profile and level information for the content referred by it. It shall be used to gain initial access to ISO/IEC 14496 content (see 8.7.3).

`objectDescriptorId` – This syntax element uniquely identifies the `ObjectDescriptor` within its name scope. The value 0 is forbidden and the value 1023 is reserved.

`URL_Flag` – a flag that indicates the presence of a `URLstring`.

`includeInlineProfileLevelFlag` – a flag that, if set to one, indicates that the subsequent profile indications take into account the resources needed to process any content that might be inlined.

`URL length` – the length of the subsequent `URLstring` in bytes.

`URLstring[]` – A string with a UTF-8 [3] encoded URL that shall point to another `InitialObjectDescriptor`. Only the content of this object descriptor shall be returned by the delivery entity upon access to this URL. Within the current name scope, the new object descriptor shall be referenced by the `objectDescriptorId` of the object descriptor carrying the `URLstring`. On name scopes see 8.7.2.4. Permissible URLs may be constrained by profile and levels as well as by specific delivery layers.

`ODProfileLevelIndication` – an indication as defined in Table 3 of the object descriptor profile and level required to process the content associated with this `InitialObjectDescriptor`.

Table 3 - ODProfileLevelIndication Values

Value	Profile	Level
0x00	Forbidden	-
0x01-0x7F	reserved for ISO use	-
0x80-0xFD	user private	-
0xFE	no OD profile specified	-
0xFF	no OD capability required	-
NOTE — Usage of the value 0xFE indicates that the content described by this <code>InitialObjectDescriptor</code> does not comply to any OD profile specified in ISO/IEC 14496-1. Usage of the value 0xFF indicates that none of the OD profile capabilities are required for this content.		

`sceneProfileLevelIndication` – an indication as defined in Table 4 of the scene graph profile and level required to process the content associated with this `InitialObjectDescriptor`.

Table 4 - sceneProfileLevelIndication Values

Value	Profile	Level
0x00	Reserved for ISO use	-
0x01	Simple2D profile	L1
0x02-0x7F	reserved for ISO use	-
0x80-0xFD	user private	-
0xFE	no scene graph profile specified	-
0xFF	no scene graph capability required	-
NOTE — Usage of the value 0xFE indicates that the content described by this <code>InitialObjectDescriptor</code> does not comply to any scene graph profile specified in ISO/IEC 14496-1. Usage of the value 0xFF indicates that none of the scene graph profile capabilities are required for this content.		

`audioProfileLevelIndication` – an indication as defined in Table 5 of the audio profile and level required to process the content associated with this `InitialObjectDescriptor`.

Table 5 - audioProfileLevelIndication Values

Value	Profile	Level
0x00	Reserved for ISO use	-
0x01	Main Profile	L1
0x02	Main Profile	L2
0x03	Main Profile	L3
0x04	Main Profile	L4
0x05	Scalable Profile	L1
0x06	Scalable Profile	L2
0x07	Scalable Profile	L3
0x08	Scalable Profile	L4
0x09	Speech Profile	L1
0x0A	Speech Profile	L2
0x0B	Synthesis Profile	L1
0x0C	Synthesis Profile	L2
0x0D	Synthesis Profile	L3
0x0E-0x7F	reserved for ISO use	-
0x80-0xFD	user private	-
0xFE	no audio profile specified	-
0xFF	no audio capability required	-

NOTE — Usage of the value 0xFE indicates that the content described by this InitialObjectDescriptor does not comply to any audio profile specified in ISO/IEC 14496-3. Usage of the value 0xFF indicates that none of the audio profile capabilities are required for this content.

visualProfileLevelIndication – an indication as defined in Table 6 of the visual profile and level required to process the content associated with this InitialObjectDescriptor.

Table 6 - visualProfileLevelIndication Values

Value	Profile	Level
0x00	Reserved for ISO use	-
0x01	Simple	L3
0x02	Simple	L2
0x03	Simple	L1
0x04	Simple Scalable	L2
0x05	Simple Scalable	L1
0x06	Core	L2
0x07	Core	L1
0x08	Main	L4
0x09	Main	L3
0x0A	Main	L2
0x0B	N-Bit	L2
0x0C	Hybrid	L2
0x0D	Hybrid	L1
0x0E	Basic Animated Texture	L2
0x0F	Basic Animated Texture	L1
0x10	Scalable Texture	L3
0x11	Scalable Texture	L2
0x12	Scalable Texture	L1
0x13	Simple Face Animation	L2
0x14	Simple Face Animation	L1
0x15-0x7F	reserved for ISO use	-
0x80-0xFD	user private	-

0xFE	no visual profile specified	-
0xFF	no visual capability required	-
NOTE — Usage of the value 0xFE indicates that the content described by this InitialObjectDescriptor does not comply to any visual profile specified in ISO/IEC 14496-2. Usage of the value 0xFF indicates that none of the visual profile capabilities are required for this content.		

graphicsProfileLevelIndication – an indication as defined in Table 7 of the graphics profile and level required to process the content associated with this InitialObjectDescriptor.

Table 7 - graphicsProfileLevelIndication Values

Value	Profile	Level
0x00	Reserved for ISO use	
0x01	Simple2D profile	L1
0x02-0x7F	reserved for ISO use	
0x80-0xFD	user private	
0xFE	no graphics profile specified	
0xFF	no graphics capability required	
NOTE — Usage of the value 0xFE may indicate that the content described by this InitialObjectDescriptor does not comply to any conformance point specified in ISO/IEC 14496-1. Usage of the value 0xFF indicates that none of the graphics profile capabilities are required for this content.		

ESD[] – an array of **ES_Descriptors** as defined in 8.6.4. The array shall have any number of one up to 30 elements.

ociDescr[] – an array of **OCI_Descriptors** as defined in 8.6.17.2 that relates to the set of audio-visual objects that are described by this initial object descriptor. The array shall have any number of zero up to 255 elements.

ipmpDescrPtr[] – an array of **IPMP_DescriptorPointer**, as defined in 8.6.12, that points to the **IPMP_Descriptors** related to the elementary stream(s) described by this object descriptor. The array shall have any number of zero up to 255 elements.

extDescr[] – an array of **ExtensionDescriptors** as defined in 8.6.15. The array shall have any number of zero up to 255 elements.

8.6.4 ES_Descriptor

8.6.4.1 Syntax

```

class ES_Descriptor extends BaseDescriptor : bit(8) tag=ES_DescrTag {
    bit(16) ES_ID;
    bit(1) streamDependenceFlag;
    bit(1) URL_Flag;
    const bit(1) reserved=1;
    bit(5) streamPriority;
    if (streamDependenceFlag)
        bit(16) dependsOn_ES_ID;
    if (URL_Flag) {
        bit(8) URLlength;
        bit(8) URLstring[URLlength];
    }
    DecoderConfigDescriptor decConfigDescr;
    SLConfigDescriptor slConfigDescr;
    IPI_DescriptorPointer ipiPtr[0 .. 1];
    IP_IdentificationDataSet ipIDS[0 .. 255];
    IPMP_DescriptorPointer ipmpDescrPtr[0 .. 255];
    LanguageDescriptor langDescr[0 .. 255];
    QoS_Descriptor qosDescr[0 .. 1];
    RegistrationDescriptor regDescr[0 .. 1];
    ExtensionDescriptor extDescr[0 .. 255];
}

```