

INTERNATIONAL STANDARD



**Consumer terminal function for access to IPTV and open internet multimedia services –
Part 5-1: Declarative application environment**

IECNORM.COM : Click to view the full PDF of IEC 62766-5-1:2017



THIS PUBLICATION IS COPYRIGHT PROTECTED

Copyright © 2017 IEC, Geneva, Switzerland

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 IEC or IEC's member National Committee in the country of the requester. If you have any questions about IEC copyright or have an enquiry about obtaining additional rights to this publication, please contact the address below or your local IEC member National Committee for further information.

IEC Central Office
3, rue de Varembe
CH-1211 Geneva 20
Switzerland

Tel.: +41 22 919 02 11
Fax: +41 22 919 03 00
info@iec.ch
www.iec.ch

About the IEC

The International Electrotechnical Commission (IEC) is the leading global organization that prepares and publishes International Standards for all electrical, electronic and related technologies.

About IEC publications

The technical content of IEC publications is kept under constant review by the IEC. Please make sure that you have the latest edition, a corrigenda or an amendment might have been published.

IEC Catalogue - webstore.iec.ch/catalogue

The stand-alone application for consulting the entire bibliographical information on IEC International Standards, Technical Specifications, Technical Reports and other documents. Available for PC, Mac OS, Android Tablets and iPad.

IEC publications search - www.iec.ch/searchpub

The advanced search enables to find IEC publications by a variety of criteria (reference number, text, technical committee,...). It also gives information on projects, replaced and withdrawn publications.

IEC Just Published - webstore.iec.ch/justpublished

Stay up to date on all new IEC publications. Just Published details all new publications released. Available online and also once a month by email.

Electropedia - www.electropedia.org

The world's leading online dictionary of electronic and electrical terms containing 20 000 terms and definitions in English and French, with equivalent terms in 16 additional languages. Also known as the International Electrotechnical Vocabulary (IEV) online.

IEC Glossary - std.iec.ch/glossary

65 000 electrotechnical terminology entries in English and French extracted from the Terms and Definitions clause of IEC publications issued since 2002. Some entries have been collected from earlier publications of IEC TC 37, 77, 86 and CISPR.

IEC Customer Service Centre - webstore.iec.ch/csc

If you wish to give us your feedback on this publication or need further assistance, please contact the Customer Service Centre: csc@iec.ch.

IECNORM.COM : Click to view the full text of IEC 60366-51:2017

INTERNATIONAL STANDARD



**Consumer terminal function for access to IPTV and open internet multimedia services –
Part 5-1: Declarative application environment**

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

ICS 33.170 35.240.95

ISBN 978-2-8322-4573-6

Warning! Make sure that you obtained this publication from an authorized distributor.

CONTENTS

FOREWORD.....	12
INTRODUCTION.....	14
1 Scope.....	15
2 Normative references	15
3 Terms, definitions and abbreviated terms	17
3.1 Terms and definitions.....	17
3.2 Abbreviated terms.....	19
4 DAE overview	19
4.1 General.....	19
4.2 Architecture of the DAE	20
4.3 Gateway discovery and control	21
4.4 Application definition.....	22
4.4.1 General	22
4.4.2 Similarities between applications and traditional web pages	22
4.4.3 Differences between applications and traditional web pages.....	22
4.4.4 The application tree	23
4.4.5 The application display model.....	23
4.4.6 The security model	23
4.4.7 Inheritance of permissions	24
4.4.8 Privileged application APIs	24
4.4.9 Active applications list	24
4.4.10 Widgets	24
4.4.11 Origin for broadcast-delivered documents.....	25
4.5 Resource management	25
4.5.1 General	25
4.5.2 Application lifecycle issues	25
4.5.3 Caching of application files.....	26
4.5.4 Memory usage.....	26
4.5.5 Instantiating embedded objects and claiming scarce system resources	26
4.5.6 Media control.....	26
4.5.7 Use of the display.....	27
4.5.8 Cross-application event handling	28
4.5.9 Tuner resources	29
4.6 Parental access control.....	30
4.7 Content download	31
4.7.1 General	31
4.7.2 Download manager.....	31
4.7.3 Content access download descriptor.....	31
4.7.4 Triggering a download	31
4.7.5 Download protocol(s).....	32
4.8 Streaming CoD	33
4.8.1 General	33
4.8.2 Unicast streaming.....	33
4.9 Scheduled content	34
4.9.1 General	34
4.9.2 Conveyance of channel list.....	34
4.9.3 Conveyance of channel list and list of scheduled recordings.....	35

4.10	DLNA RUI remote control function	36
4.10.1	General	36
4.10.2	Interfaces used by the DLNA RUI remote control function	37
4.11	Power consumption	38
4.11.1	General	38
4.11.2	DAE application wake-up support	39
4.11.3	OITF hibernate support	40
4.11.4	State diagram for the power state	41
4.12	Display model	41
5	DAE application model	41
5.1	Application lifecycle	41
5.1.1	General	41
5.1.2	Creating a new application	41
5.1.3	Stopping an application	43
5.1.4	Application boundaries	43
5.2	Application announcement and signalling	43
5.2.1	Overview	43
5.2.2	General	44
5.2.3	Broadcast-related applications	45
5.2.4	Service provider related applications	50
5.2.5	Broadcast-independent applications	51
5.2.6	Switching between applications	51
5.2.7	Signalling format	51
5.2.8	Widgets lifecycle	55
5.3	Event notifications	56
5.3.1	General	56
5.3.2	Event notification framework based on CEA 2014	57
5.3.3	IMS event notification framework	59
6	Formats	66
6.1	Web standards TV profile	66
6.1.1	General	66
6.1.2	Additional restrictions and requirements	67
6.2	Still image formats	67
6.3	Media formats	67
6.3.1	General	67
6.3.2	Media format of A/V media except for audio from memory	67
6.3.3	Media format of A/V media for audio from memory	68
6.3.4	Media transport	68
6.4	SVG	68
7	APIs	68
7.1	Object factory API	68
7.1.1	General	68
7.1.2	Methods	69
7.1.3	Examples	71
7.2	Application management APIs	72
7.2.1	General	72
7.2.2	The application/oipfApplicationManager embedded object	72
7.2.3	Application class	76
7.2.4	The ApplicationCollection class	79

7.2.5	The ApplicationPrivateData class	79
7.2.6	The Keyset class	80
7.2.7	New DOM events for application support	82
7.2.8	Examples	83
7.2.9	Widget APIs	84
7.3	Configuration and setting APIs	85
7.3.1	General	85
7.3.2	The application/oipfConfiguration embedded object	85
7.3.3	The Configuration class	86
7.3.4	The LocalSystem class	88
7.3.5	The NetworkInterface class	94
7.3.6	The AVOutput class	94
7.3.7	The NetworkInterfaceCollection class	98
7.3.8	The AVOutputCollection class	98
7.3.9	The TunerCollection class	98
7.3.10	The Tuner class	98
7.3.11	The SignalInfo class	99
7.3.12	The LNBInfo class	100
7.3.13	The StartupInformation class	101
7.4	Content download APIs	101
7.4.1	General	101
7.4.2	The application/oipfDownloadTrigger embedded object	101
7.4.3	Extensions to application/oipfDownloadTrigger	104
7.4.4	The application/oipfDownloadManager embedded object	104
7.4.5	The Download class	110
7.4.6	The DownloadCollection class	113
7.4.7	The DRMControlInformation class	113
7.4.8	The DRMControlInfoCollection class	114
7.5	Content on demand metadata APIs	114
7.5.1	General	114
7.5.2	The application/oipfCodManager embedded object	114
7.5.3	The ContentCatalogueCollection class	116
7.5.4	The ContentCatalogue class	116
7.5.5	The ContentCatalogueEvent class	117
7.5.6	The CODFolder class	117
7.5.7	The CODAsset class	118
7.5.8	The CODService class	121
7.6	Content service protection API	123
7.6.1	General	123
7.6.2	The application/oipfDrmAgent embedded object	123
7.7	Gateway discovery and control APIs	125
7.7.1	General	125
7.7.2	The application/oipfGatewayInfo embedded object	126
7.8	Communication services APIs	128
7.8.1	General	128
7.8.2	The application/oipfCommunicationServices embedded object	129
7.8.3	Extensions to application/oipfCommunicationServices for presence and messaging services	132
7.8.4	The UserData class – Properties	135

7.8.5	The UserDataCollection class	136
7.8.6	The FeatureTag class – Properties	136
7.8.7	The FeatureTagCollection class	136
7.8.8	The Contact class – Properties	136
7.8.9	The ContactCollection class	136
7.8.10	Extensions to application/oipfCommunicationServices for voice telephony services	137
7.8.11	Extensions to application/oipfCommunicationServices for video telephony services	143
7.8.12	The DeviceInfo class	145
7.8.13	The DeviceInfoCollection class	146
7.8.14	The CodecInfo class	146
7.8.15	The CodecInfoCollection class	146
7.9	Parental rating and parental control APIs	147
7.9.1	General	147
7.9.2	The application/oipfParentalControlManager embedded object	147
7.9.3	The ParentalRatingScheme class	150
7.9.4	The ParentalRatingSchemeCollection class	151
7.9.5	The ParentalRating class	152
7.9.6	The ParentalRatingCollection class	154
7.10	Scheduled Recording APIs	155
7.10.1	General	155
7.10.2	The application/oipfRecordingScheduler embedded object	155
7.10.3	The ScheduledRecording class	158
7.10.4	The ScheduledRecordingCollection class	162
7.10.5	Extension to application/oipfRecordingScheduler for control of recordings	162
7.10.6	The Recording class	164
7.10.7	The RecordingCollection class	167
7.10.8	The PVREvent class	167
7.10.9	The Bookmark class	167
7.10.10	The BookmarkCollection class	168
7.11	Remote Management APIs	168
7.11.1	General	168
7.11.2	The application/oipfRemoteManagement embedded object	168
7.12	Metadata APIs	172
7.12.1	General	172
7.12.2	The application/oipfSearchManager embedded object	173
7.12.3	The MetadataSearch class	175
7.12.4	The Query class	180
7.12.5	The SearchResults class	181
7.12.6	The MetadataSearchEvent class	182
7.12.7	The MetadataUpdateEvent class	182
7.13	Scheduled content and hybrid tuner APIs	182
7.13.1	General	182
7.13.2	The video/broadcast embedded object	182
7.13.3	Extensions to video/broadcast for recording and time-shift	198
7.13.4	Extensions to video/broadcast for access to EIT p/f	207
7.13.5	Extensions to video/broadcast for playback of selected components	208
7.13.6	Extensions to video/broadcast for parental ratings errors	209

7.13.7	Extensions to video/broadcast for DRM rights errors.....	210
7.13.8	Extensions to video/broadcast for current channel information.....	211
7.13.9	Extensions to video/broadcast for creating channel lists from SD&S fragments	211
7.13.10	The ChannelConfig class	211
7.13.11	The ChannelList class	216
7.13.12	The Channel class	217
7.13.13	The FavouriteListCollection class	222
7.13.14	The FavouriteList class	223
7.13.15	Extensions to video/broadcast for channel scan.....	225
7.13.16	The ChannelScanEvent class	225
7.13.17	The ChannelScanOptions class	225
7.13.18	The ChannelScanParameters class	225
7.13.19	The DVBTChannelScanParameters class	225
7.13.20	The DVBSChannelScanParameters class	227
7.13.21	The DVBCChannelScanParameters class.....	228
7.13.22	Extensions to video/broadcast for synchronization.....	229
7.13.23	The ATSCChannelScanParameters class	230
7.14	Media playback APIs.....	231
7.14.1	General	231
7.14.2	The A/V Control object.....	231
7.14.3	Extensions to A/V Control object for playback through Content-Access Streaming Descriptor	238
7.14.4	Extensions to A/V Control object for trickmodes.....	239
7.14.5	Extensions to A/V Control object for playback of selected components	240
7.14.6	Extensions to A/V Control object for parental rating errors	240
7.14.7	Extensions to A/V Control object for DRM rights errors	242
7.14.8	Extensions to A/V Control object for playing media objects	243
7.14.9	Extensions to A/V Control object for UI feedback of buffering A/V content	243
7.14.10	DOM events for A/V Control object	247
7.14.11	Playback of memory audio	248
7.14.12	Extensions to A/V Control object for media queuing.....	250
7.14.13	Extensions to A/V Control object for volume control.....	251
7.14.14	Extensions to A/V Control object for resource management.....	251
7.15	Miscellaneous APIs.....	252
7.15.1	The application/oipfMDTF embedded object	252
7.15.2	The application/oipfStatusView embedded object	254
7.15.3	The application/oipfCapabilities embedded object.....	255
7.15.4	The Navigator class	256
7.15.5	Debug print API	256
7.16	Shared Utility classes and features	256
7.16.1	Base collections	256
7.16.2	The Programme class	257
7.16.3	The ProgrammeCollection class	262
7.16.4	The DiscInfo class	262
7.16.5	Extensions for playback of selected media components.....	262
7.16.6	Additional support for protected content.....	266
7.17	DLNA RUI remote control function APIs	267
7.17.1	General	267

7.17.2	The application/oipfRemoteControlFunction embedded object	267
8	System integration aspects.....	272
8.1	HTTP protocol.....	272
8.1.1	General	272
8.1.2	HTTP User-Agent header	272
8.1.3	HTTP X-OITF-RCF-User-Agent header	273
8.2	Mapping from APIs to protocols	273
8.2.1	General	273
8.2.2	CoD download over HTTP	274
8.2.3	CoD unicast streaming with SIP session management.....	274
8.2.4	Scheduled content multicast streaming with SIP session management.....	278
8.2.5	Communication services with SIP session management	284
8.2.6	CoD unicast streaming over RTP and HTTP	284
8.2.7	Scheduled content multicast streaming.....	288
8.3	URI schemes and their usage	289
8.3.1	General	289
8.3.2	Media fragments support	290
8.4	Mapping from APIs to content formats	291
8.4.1	Character conversion.....	291
8.4.2	AVComponent	291
8.4.3	Channel.....	294
8.4.4	Programme, ScheduledRecording, Recording and Download.....	299
8.4.5	Exposing audio description streams as AVComponent objects.....	307
8.4.6	HTML5 media element mapping.....	307
8.5	DLNA RUI remote control function implementation	309
8.5.1	General	309
8.5.2	Relationship between DAE application and control UI	309
8.5.3	XML UI listing provisioning	310
8.5.4	Retrieving the control UI	312
8.5.5	Receiving and responding to a message between the control UI in the remote control device and OITF.....	313
8.5.6	Notification to the remote control device	315
8.5.7	Handling multiple DAE applications and multiple remote control devices	316
9	Capabilities.....	317
9.1	Minimum DAE capability requirements	317
9.1.1	General	317
9.1.2	SSL/TTLS Requirements	320
9.2	Default UI profiles	321
9.3	Client capability description	324
9.3.1	General	324
9.3.2	Tuner/broadcast capability indication	325
9.3.3	Broadcast content over IP capability indication	326
9.3.4	PVR capability indication	326
9.3.5	Download CoD capability indication	327
9.3.6	Parental ratings	328
9.3.7	Extended A/V API support	329
9.3.8	OITF metadata API support	329
9.3.9	OITF configuration API support.....	329
9.3.10	Communication services API Support	330

9.3.11	DRM capability indication	330
9.3.12	Media profile capability indication	331
9.3.13	Remote diagnostics support.....	332
9.3.14	SVG	332
9.3.15	Third party notification support	333
9.3.16	Multicast delivery terminating function support.....	333
9.3.17	Other capability extensions.....	333
9.3.18	HTML5 video	333
9.3.19	DLNA RUI remote control function support	333
9.3.20	Power consumption	333
9.3.21	Widgets	333
9.3.22	Buffer control of AV content playback API support.....	334
9.3.23	Temporal clipping	334
9.3.24	Capability elements from other schemas.....	335
9.3.25	Pointer support	335
10	Security.....	335
10.1	Application / service security.....	335
10.1.1	General	335
10.1.2	OITF requirements.....	335
10.1.3	Server requirements	336
10.1.4	Specific security requirements for privileged JavaScript APIs	336
10.1.5	Permission names	339
10.1.6	Loading documents from different domains.....	341
10.2	User authentication.....	341
10.3	DLNA RUI remote control.....	341
11	DAE Widgets	341
11.1	General.....	341
11.2	Widgets packaging and configuration.....	341
11.3	Access request	342
11.4	Widget interface.....	342
11.5	Digital signature.....	342
12	Graphics performance	343
12.1	Overview.....	343
12.2	Performance levels	343
12.3	Minimum 2D graphics performance	343
12.4	Minimum 3D graphics performance	344
12.5	Minimum canvas performance.....	344
12.6	Minimum WebGL performance	344
12.7	Performance measurement	344
Annex A	(informative) Design rationale – application model	346
Annex B	(informative) Clarification of download CoD, streaming CoD and CSP interfaces.....	347
B.1	Overview.....	347
B.2	List of interfaces	348
B.2.1	Interface a): browse, select and purchase content from CoD store.....	348
B.2.2	Interface b*): in-session interaction from web page with underlying DRM agent	348
B.2.3	Interface c*): autonomous out-of-session interaction between DRM agent and CoD store.....	349

B.2.4	Interface d*): downloading content.....	349
B.2.5	Interface e*): unicast streaming and playback of downloaded content using A/V Control object	351
B.2.6	Interface f): request licence	351
B.2.7	Interface g*): local metadata based applications	351
B.3	Additional notes about content-on-demand	351
Annex C (normative)	Content access descriptor syntax and semantics	352
C.1	Content access download descriptor format	352
C.2	Content access streaming descriptor format.....	353
C.3	Abstract content access descriptor format.....	354
Annex D (normative)	Capability extensions schema	359
Annex E (normative)	Client channel listing format	362
Annex F (normative)	Display model.....	366
F.1	Logical plane model.....	366
F.2	Interaction with the video/broadcast and A/V Control objects	367
F.3	Graphic safe area	368
F.4	Current channel	368
Annex G (normative)	Backwards compatible profile of HTML5 media elements	370
G.1	General.....	370
G.2	Video element.....	370
G.3	Audio element.....	370
G.4	Source element.....	371
G.5	Media element	371
G.6	Other object types.....	372
G.7	Dependencies	372
Annex H (informative)	DLNA RUI remote control function sequences	373
H.1	Remote UI and box models.....	373
H.1.1	Overview	373
H.1.2	i-box model.....	374
H.1.3	2-box model.....	374
H.1.4	3-box model.....	375
H.2	DLNA RUI remote control function sequences.....	375
H.2.1	General	375
H.2.2	Launching a DAE application to obtain the Control UI	376
H.2.3	Obtaining the control UI from a running DAE application.....	378
H.2.4	Sending and receiving messages between the remote control device and DAE application	380
Annex I (normative)	Collections	382
I.1	General.....	382
I.2	The Collection template	382
I.2.1	General	382
I.2.2	Properties.....	382
I.2.3	Methods	382
Annex J (informative)	SVG video tag support.....	383
Annex K (informative)	Multimedia telephony sequences.....	386
K.1	General.....	386
K.2	Full-duplex voice telephony call flow	386
K.3	Full-duplex video telephony call flow	388

K.4	Capture device and call parameters setting flow	389
K.5	Full-duplex Voice to Video telephony session update flow.....	390
Annex L (informative)	Server root certificate selection policy	392
L.1	Overview.....	392
L.2	Background.....	392
L.3	Policy.....	392
Annex M (normative)	Changes to 5.6.2 of CEA-2014-A	394
Bibliography		397
Figure 1	– OITF architecture	20
Figure 2	– OIPF architecture with DLNA RUI RCF scenario	37
Figure 3	– State diagram of OITF power states	41
Figure 4	– Behaviour when the selected channel changes	47
Figure 5	– Behaviour when the application signalling for the currently selected channel changes or when a running broadcast-related application exits	49
Figure 6	– General event notification architecture on OITF and remote UI server	57
Figure 7	– HNI-IGI transaction for outgoing SIP requests from a DAE application	60
Figure 8	– HNI-IGI transaction for in-session incoming SIP request	62
Figure 9	– What happens when the OITF is first turned on	64
Figure 10	– User logs in using the DAE interface	65
Figure 11	– Unsolicited message from the network	66
Figure 12	– State diagram for embedded application/oipfDownloadManager objects	105
Figure 13	– State machine for a metadata search	176
Figure 14	– State diagram for embedded video/broadcast objects	183
Figure 15	– PVR States for recording and timeshifting using video/broadcast	199
Figure 16	– State diagram for embedded A/V Control objects (normative).....	236
Figure 17	– XML UI listing provisioning	310
Figure 18	– Remote control message handling.....	313
Figure 19	– Remote control device changes mapping between DAE applications	317
Figure 20	– Remote control device retains control of DAE application	317
Figure B.1	– Main scenario	347
Figure F.1	– Logical plane model	366
Figure F.2	– Graphic safe area	368
Figure H.1	– i-box model.....	374
Figure H.2	– 2-box Model.....	375
Figure H.3	– 3-box model.....	375
Figure H.4	– Launching of a DAE application	377
Figure H.5	– Obtaining remote control of a running DAE application	379
Figure H.6	– Message flow between the remote control device and the DAE application	381
Figure K.1	– Full-duplex voice telephony call flow	387
Figure K.2	– Full-duplex Video telephony call flow	388
Figure K.3	– Capture device and call parameters setting flow	390
Figure K.4	– Full-duplex Voice to Video telephony session update flow.....	391

Table 1 – Events applicable for cross application event handling	29
Table 2 – Application signalling.....	51
Table 3 – DAE application control codes	53
Table 4 – Supported application signalling features	53
Table 5 – Key to status column	55
Table 6 – New DOM events for application support.....	83
Table 7 – Metadata search states	177
Table 8 – State transitions for the video/broadcast embedded object	184
Table 9 – Properties of the A/V Control object when the type attribute refers to video or audio	232
Table 10 – Additional properties of the A/V Control object when the type attribute refers to video.....	233
Table 11 – Methods of the A/V Control object when the type attribute refers to video or audio	234
Table 12 – Additional methods of the A/V Control object when the type attribute refers to video.....	234
Table 13 – Additional applicable requirements from CEA-2014	235
Table 14 – URI schemes and usages	290
Table 15 – Base UI profile names	321
Table 16 – Complementary UI profile name fragments	322
Table 17 – Minimum 2D graphics performance.....	344
Table F.1 – Clarification of the "current channel" concept in different scenarios	369
Table J.1 – SVG video tag support.....	383

INTERNATIONAL ELECTROTECHNICAL COMMISSION

CONSUMER TERMINAL FUNCTION FOR ACCESS TO IPTV AND OPEN INTERNET MULTIMEDIA SERVICES –

Part 5-1: Declarative application environment

FOREWORD

- 1) The International Electrotechnical Commission (IEC) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, IEC publishes International Standards, Technical Specifications, Technical Reports, Publicly Available Specifications (PAS) and Guides (hereafter referred to as "IEC Publication(s)"). Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation. IEC collaborates closely with the International Organization for Standardization (ISO) in accordance with conditions determined by agreement between the two organizations.
- 2) The formal decisions or agreements of IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC National Committees.
- 3) IEC Publications have the form of recommendations for international use and are accepted by IEC National Committees in that sense. While all reasonable efforts are made to ensure that the technical content of IEC Publications is accurate, IEC cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.
- 4) In order to promote international uniformity, IEC National Committees undertake to apply IEC Publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC Publication and the corresponding national or regional publication shall be clearly indicated in the latter.
- 5) IEC itself does not provide any attestation of conformity. Independent certification bodies provide conformity assessment services and, in some areas, access to IEC marks of conformity. IEC is not responsible for any services carried out by independent certification bodies.
- 6) All users should ensure that they have the latest edition of this publication.
- 7) No liability shall attach to IEC or its directors, employees, servants or agents including individual experts and members of its technical committees and IEC National Committees for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication, use of, or reliance upon, this IEC Publication or any other IEC Publications.
- 8) Attention is drawn to the Normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.
- 9) Attention is drawn to the possibility that some of the elements of this IEC Publication may be the subject of patent rights. IEC shall not be held responsible for identifying any or all such patent rights.

International Standard IEC 62766-5-1 has been prepared by IEC technical committee 100: Audio, video and multimedia systems and equipment.

The text of this standard is based on the following documents:

CDV	Report on voting
100/2548/CDV	100/2662/RVC

Full information on the voting for the approval of this standard can be found in the report on voting indicated in the above table.

This publication has been drafted in accordance with the ISO/IEC Directives, part 2.

A list of all parts in the IEC 62766 series, published under the general title *Consumer terminal function for access to IPTV and open internet multimedia services*, can be found on the IEC website.

In this standard, the following print type is used: object and event labels: Lucida Console.

The committee has decided that the contents of this publication will remain unchanged until the stability date indicated on the IEC web site under "<http://webstore.iec.ch>" in the data related to the specific publication. At this date, the publication will be

- reconfirmed,
- withdrawn,
- replaced by a revised edition, or
- amended.

A bilingual version of this publication may be issued at a later date.

IMPORTANT – The 'colour inside' logo on the cover page of this publication indicates that it contains colours which are considered to be useful for the correct understanding of its contents. Users should therefore print this document using a colour printer.

INTRODUCTION

The IEC 62766 series is based on a series of specifications that was originally developed by the OPEN IPTV FORUM (OIPF). They specify the user-to-network interface (UNI) for consumer terminals to access IPTV and open internet multimedia services over managed or non-managed networks as defined by OIPF.

IECNORM.COM : Click to view the full PDF of IEC 62766-5-1:2017

CONSUMER TERMINAL FUNCTION FOR ACCESS TO IPTV AND OPEN INTERNET MULTIMEDIA SERVICES –

Part 5-1: Declarative application environment

1 Scope

This part of IEC 62766 specifies the Declarative Application Environment (DAE) component of the OIPF terminal function (OITF). The DAE is a declarative language based environment (browser) based on the OIPF web standards TV profile specified in IEC 62766-5-2 for the presentation of user interfaces and including scripting support for interaction with network server-side applications and access to the APIs of the other OITF functions.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

IEC 62481, *Digital living network alliance*

IEC 62766-1, *Open IPTV Forum (OIPF) consumer terminal function and network interfaces for access to IPTV and open Internet multimedia services – Part 1: General*

IEC 62766-2-1, *Open IPTV Forum (OIPF) consumer terminal function and network interfaces for access to IPTV and open Internet multimedia services – Part 2-1: Media Formats*

IEC 62766-2-2, *Open IPTV Forum (OIPF) consumer terminal function and network interfaces for access to IPTV and open Internet multimedia services – Part 2-2: HTTP Adaptive Streaming*

IEC 62766-3:2016, *Open IPTV Forum (OIPF) consumer terminal function and network interfaces for access to IPTV and open Internet multimedia services – Part 3: Content Metadata*

IEC 62766-4-1:2017, *Open IPTV Forum (OIPF) consumer terminal function and network interfaces for access to IPTV and open Internet multimedia services – Part 4-1: Protocols*

IEC 62766-5-2, *Open IPTV Forum (OIPF) consumer terminal function and network interfaces for access to IPTV and open Internet multimedia services – Part 5-2: Web standards TV profile*

IEC 62766-7:2017, *Open IPTV Forum (OIPF) consumer terminal function and network interfaces for access to IPTV and open Internet multimedia services – Part 7: Authentication, content protection and service protection*

ISO/IEC 15938-5:2003, *Multimedia content description interface – Part 5: Multimedia description schemes*

ISO/IEC 15948:2004, *Information technology – Computer graphics and image processing – Portable Network Graphics (PNG): Functional specification*

ISO/IEC 23009-1:2014, *Information technology – Dynamic adaptive streaming over HTTP (DASH) – Part 1: Media presentation description and segment formats*

ISO 639-2, *Codes for the representation of names of languages – Part 2: Alpha-3 code*

ETSI TS 102 034 V1.5.1, *Digital Video Broadcasting (DVB); Transport of MPEG-2 TS Based DVB Services over IP Based Networks (and associated XML)*

ETSI TS 102 323, *Digital Video Broadcasting (DVB); Carriage and signalling of TV Anytime information in DVB transport streams*

ETSI TS 102 539, *Digital Video Broadcasting (DVB); Carriage of Broadband Content Guide (BCG) information over Internet Protocol (IP)*

ETSI TS 102 809, *Digital Video Broadcasting (DVB); Signalling and carriage of interactive applications and services in Hybrid broadcast/broadband environments*

ETSI TS 102 851, *Digital Video Broadcasting (DVB); Uniform Resource Identifiers (URI) for DVB Systems*

ETSI TS 183 063, *Telecommunications and Internet converged Services and Protocols for Advanced Networking (TISPAN); IMS-based IPTV stage 3 specification*

ETSI EN 300 468, *Digital Video Broadcasting (DVB); Specification for Service Information (SI) in DVB Systems*

IETF RFC 1321, *The MD5 Message-Digest Algorithm*

IETF RFC 1918, *Address Allocation for Private Internets*

IETF RFC 2246, *The Transport Layer Security (TLS) Protocol Version 1.0*

IETF RFC 2326, *Real Time Streaming Protocol (RTSP)*

IETF RFC 2616, *Hypertext Transfer Protocol – HTTP/1.1*

IETF RFC 2818, *HTTP over TLS*

IETF RFC 3550, *RTP: A Transport Protocol for Real-Time Applications*

IETF RFC 3840, *Indicating User Agent Capabilities in the Session Initiation Protocol (SIP)*

IETF RFC 3841, *Caller Preferences for the Session Initiation Protocol (SIP)*

IETF RFC 5019, *The Lightweight Online Certificate Status Protocol (OCSP) Profile for High-Volume Environments*

IETF RFC 5246, *The Transport Layer Security (TLS) Protocol Version 1.2*

IETF RFC 5280, *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*

IETF RFC 5746, *Transport Layer Security (TLS) Renegotiation Indication Extension*

IETF RFC 6265, *HTTP State Management Mechanism*

IETF RFC 6454, *The Web Origin Concept*, December 2011

3GPP TS 24.229, *IP Multimedia Call Control Protocol based on Session Initiation Protocol (SIP) and Session Description Protocol (SDP) Stage 3 (Release 8)*

3GPP TS 26.234 V9.3.0 (2010-06), *Transparent end-to-end Packet-switched Streaming Service (PSS) Protocols and codecs (Release 9)*

Consumer Electronics Association CEA-2014-A (including the August 2008 Errata), *Web-based Protocol Framework for Remote User Interface on UPnP Networks and the Internet (Web4CE)*

Graphics Interchange Format version 89a

Available from <http://www.w3.org/Graphics/GIF/spec-gif89a.txt>

Open Mobile Alliance, *OMA-TS-Presence_SIMPLE_XDM-V1_1-20080627-A, Presence XDM Specification*

Open Mobile Alliance, *OMA-TS-SIMPLE_IM-V1_0-20080820-D, Instant Messaging using SIMPLE*

W3C, *Web Storage*, W3C Recommendation 30 July 2013

Available at <http://www.w3.org/TR/2013/REC-webstorage-20130730/>

W3C, *Widget Access Request Policy*, W3C Recommendation 7 February 2012

Available at: <http://www.w3.org/TR/2012/REC-widgets-access-20120207/>

W3C, *Widget Interface*, W3C Recommendation 31 October 2013

Available at: <http://www.w3.org/TR/2013/REC-widgets-apis-20131031/>

W3C, *XML Digital Signatures for Widgets*, W3C Recommendation 18 April 2013

Available at: <http://www.w3.org/TR/2013/REC-widgets-digsig-20130418/>

W3C, *Packaged Web Apps (Widgets) – Packaging and XML Configuration (Second Edition)*, W3C Recommendation 27 November 2012

Available at: <http://www.w3.org/TR/2012/REC-widgets-20121127/>

W3C, *Media Fragments URI 1.0 (basic)*, W3C Recommendation 25 September 2012

Available at <http://www.w3.org/TR/2012/REC-media-frags-20120925/>

3 Terms, definitions and abbreviated terms

3.1 Terms and definitions

For the purposes of this document, the terms and definitions given in IEC 62766-1 and the following apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

- IEC Electropedia: available at <http://www.electropedia.org/>
- ISO Online browsing platform: available at <http://www.iso.org/obp>

3.1.1

audio from memory

audible notifications and audio clips intended to be played from memory

3.1.2

broadcast related application

interactive application associated with a television or radio channel, with part of a television channel (e.g. a particular program or show) or other television content

Note 1 to entry: Often referred to as “red button” applications in the industry, regardless of how they are actually started by the end user.

3.1.3

broadcast independent application

interactive application not related to any TV channel or TV content or to the currently selected service provider

3.1.4

control UI

remote UI that controls DAE applications in the OITF, sent from an IPTV applications server via the OITF or pre-stored in the OITF, and rendered in the DLNA RUIC on the remote control device

3.1.5

DLNA RUIC

DLNA device with the role of finding and loading remote UI content exposed by a DLNA RUIS capability and rendering and interacting with the UI content

Note 1 to entry: This terminology references the DLNA RUI specification (IEC 62481-6-1).

3.1.6

DLNA RUIS

DLNA function in the OITF with the role of exposing and sourcing UI content

Note 1 to entry: This terminology references the DLNA RUI specification (IEC 62481-6-1).

3.1.7

embedded object

software module that extends the capabilities of the OITF browser

Note 1 to entry: Features provided by an embedded object are made available to DAE applications through the methods and properties of a specific JavaScript¹ object.

3.1.8

HTML document

XHTML document and associated style and script files conforming to the restrictions and extensions defined in this document

3.1.9

key event

event sent to a DAE application in response to input from the end-user

¹ JavaScript is a programming language that has been standardised by ECMA as ECMAScript® and as ISO/IEC 16262. This information is given for the convenience of users of this document and does not constitute an endorsement by IEC of the product named. Equivalent products may be used if they can be shown to lead to the same results.

Note 1 to entry: This input is typically generated in response to the end-user pressing a button on a conventional remote control. It may also be generated by some other mechanism on alternative input devices such as game controllers, touch-screens, wands or drastically reduced remote controls.

3.1.10

non-visual embedded object

embedded object that has no visible representation and cannot get input focus

3.1.11

remote control device

mobile or portable device which has the functionality of the DLNA RUIC

3.1.12

remote UI

display of a UI from one device on a second (remote) device across a network

3.1.13

service provider related application

interactive application related to the service provider selected through the service provider selection process

3.1.14

trick mode

facility to allow the user to control the playback of content, such as pause, fast and slow playback, reverse playback, instant access, replay, forward and reverse skipping

3.2 Abbreviated terms

For the purposes of this document, the abbreviations given in IEC 62766-1, as well as the following apply.

AJAX	Asynchronous JavaScript and XML
CRID	Content Reference Identifier
DOM	Document Object Model
GIF	Graphics Interchange Format
HAS	HTTP Adaptive Streaming
HE-AAC	High Efficiency AAC
IR	Infra-Red
JPEG	Joint Photographic Experts Group
MPD	Media Presentation Description
PNG	Portable Network Graphics
PSI	Public Service Identifier
RCF	Remote Control Function
SVG	Scalable Vector Graphics
TLS	Transport Layer Security
WAVE	Waveform audio format

4 DAE overview

4.1 General

This document builds on a selection of W3C specifications defined in IEC 62766-5-2 with additions defined in this part of IEC 62766-5 in order to expose to an IPTV service provider the capabilities of the OITF. In clauses and subclauses of this document whose presence is

indicated by one of the capabilities defined in 9.3, the use of the terms "shall" or "required" applies only when the capability is made available to DAE applications. They do not have the effect of making that subclause mandatory. In clauses and subclauses of this document that are not covered by capabilities, terms such as "shall" apply as used in each clause and subclause.

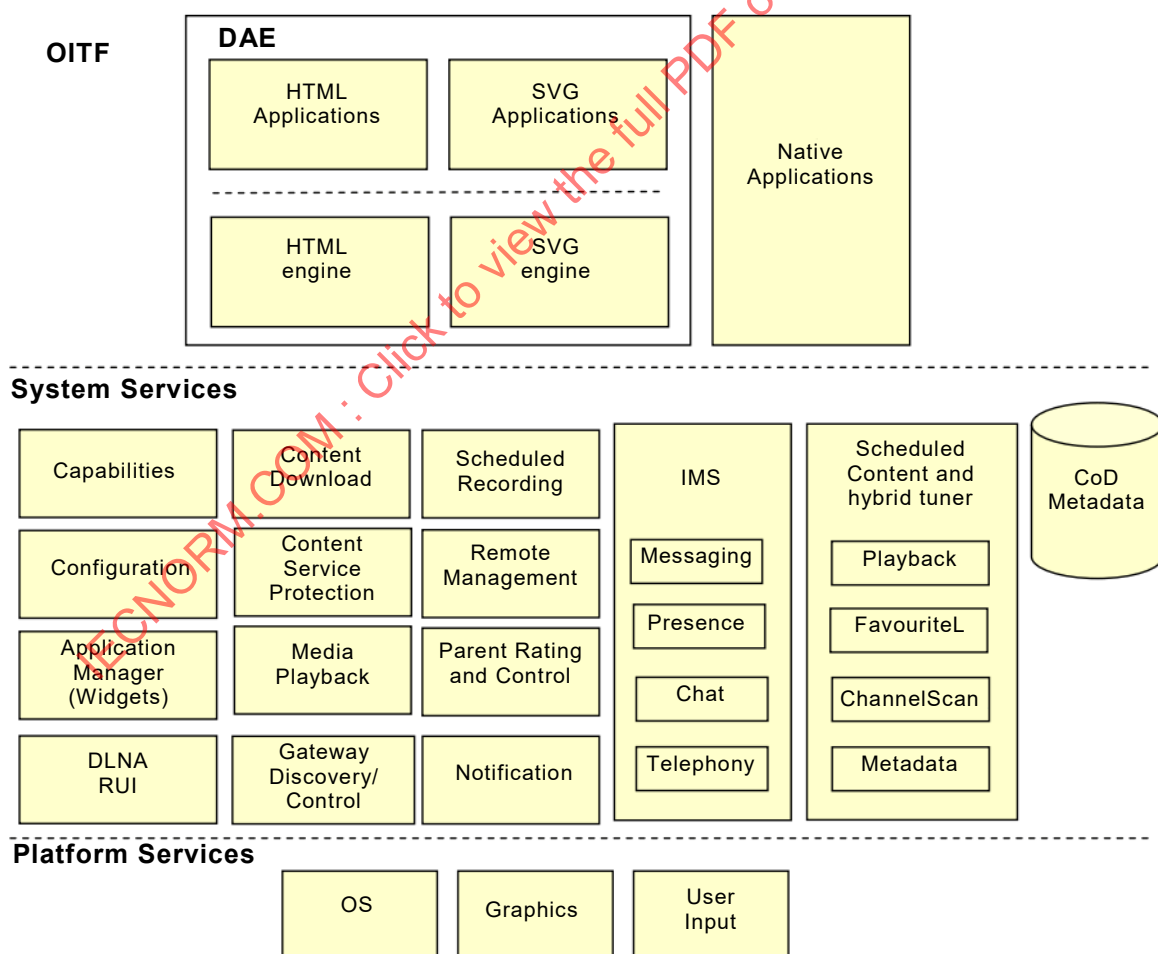
In this document, "application" means "declarative application" (browser-based application) throughout the DAE platform specification, as opposed to the "procedural applications" (Java-based applications), defined in IEC 62766-6, the PAE platform specification.

In the documented APIs, JavaScript attributes are read-write unless otherwise specified.

The type "Integer" is not a valid JavaScript type as is. It is used as a short hand notation for a subset of type "Number", which includes only the numbers that can be written without a fractional or decimal component.

4.2 Architecture of the DAE

Figure 1 provides an overview of the OITF architecture in relation to this document.



IEC

Figure 1 – OITF architecture

The various system services are described below:

Application Manager (Widgets): this service handles the starting and stopping of applications and the downloading, starting, stopping and removal of widgets. See 4.4, 5.1, 5.2.8, 7.2 and 11.

Capabilities: this service handles terminal capabilities and exposing them to applications. See 7.15.3 and 9.3.

CoD metadata: this service handles the downloading, storage and retrieval of CoD metadata. See 7.5.

Configurations: this service handles reporting and changing device configuration and power management. See 7.3.

Content Download: this service handles initiating downloading of content by applications, downloading the content and managing content once downloaded. See 4.7.2 and 7.4.

Content Service Protection: this service handles content and service protection. See 7.6.

DLNA RUI: this service enables a DAE application on an OITF to export a user interface to another device in the home as defined by the DLNA remote UI specification. See 4.10, 7.17 and 8.5.

Gateway Discovery/Control: this service handles gateways, including discovery and managing information about them. See 4.3 and 7.7.

IMS – Messaging, Presence, Chat, Telephony: this service handles IMS including messaging, presence, chat and telephony. See 7.8.

Media Playback: this service handles playback of media including streaming on-demand, downloaded content, and scheduled content that has been recorded. Live scheduled content is covered in a different service. See 4.8.2 and 7.14.

Notification: this service handles notifications from the network to the OITF. See 5.3.

Parental Rating and Control: this service handles parental rating and control including reporting of changing of parental rating status and providing parental rating PIN codes. See 4.6, 7.9, 7.13.6 and 7.14.6.

Remote Management: this service handles remote management when supported as a DAE application. See 7.11.

Scheduled Content and hybrid tuner – playback, favourite lists, channel scan, metadata: this service handles scheduled content services whether these are delivered by IP or by a classical cable, satellite or terrestrial tuner in a hybrid device. See 4.8.2.3, 4.9, 7.12 and 7.13.

Scheduled Recording: this service handles recording of scheduled content. See 7.10.

NOTE Native applications are out of scope of the DAE specification.

4.3 Gateway discovery and control

This subclause describes how DAE applications discover the information of the gateway and subsequently interacts with the gateway. The discovery of the IG and AG by the OITF are defined in 10.1 of IEC 62766-4-1:2017. The discovery takes place prior to the DAE application being initialized. The information about the discovered gateways is made available to DAE applications through the application/oipfGatewayInfo embedded object. DAE applications can use this gateway information to interact with the discovered gateways (e.g.

IG, AG, CSP Gateway and so on). The application/oipfGatewayInfo embedded object shall be made accessible through the DOM with the interface as defined in 7.7.2.

Access to the functionality of the application/oipfGatewayInfo embedded object is privileged and shall adhere to the security requirements defined in 10.1.

4.4 Application definition

4.4.1 General

This subclause defines what is meant by the concept of a DAE application, which files and assets are considered to be part of a DAE application, and how this relates to DAE application security and lifecycle.

A DAE application is either:

- an associated collection of documents (HTML or SVG where supported) from within a common boundary (see 5.1.4 for more details), or
- a Widget as specified in 4.4.10.

While the application is loaded within the browser, an additional browser object (the oipfApplicationManager object) defined in 7.2.2 is present and accessible by the DAE application. The oipfApplicationManager object provides access to the Application class defined in 7.2.3.

The difference between a DAE application and a traditional web page is that web pages are stand-alone with no formal concept of a group of pages or a context within which a group of pages are loaded and execute. For this reason, the definition and details of a DAE application focuses on the application execution environment and the additional capabilities provided to DAE applications. Subclauses 4.4.2 to 4.4.11 describe some of the differences. Additional details about the DAE application lifecycle can be found in 5.1.

4.4.2 Similarities between applications and traditional web pages

DAE applications are comprised of pages that are conceptually no different from traditional web pages. Both pages in a DAE application and traditional web pages can include the contents of other documents. These included documents can have a variety of types, including Cascading Style Sheets (CSS), JavaScript, SVG, JPEG, PNG and GIF.

A dynamic DOM, combined with XMLHttpRequest, permits AJAX-style changes to the current page in a DAE application or web page without necessarily replacing the entire document.

4.4.3 Differences between applications and traditional web pages

A DAE application provides shared context and state common to a number of pages – a concept which doesn't formally exist in the web. Loading and unloading pages within the context of a DAE application is the same as loading and unloading web pages.

The application context includes information about the state of an application from the platform's perspective – permissions, priority (for example, which to terminate first in the event of insufficient resources) and similar information that spans all documents within an application during the lifetime of that application.

An OITF may support the execution of more than one application simultaneously. Applications may share the same screen estate in a defined and controlled fashion. This differs from multiple web pages, which are typically handled through different browser "windows" or "tabs" and may not share the same screen estate concurrently (although the details of this behaviour are often browser-dependent). This also differs from the use of frames, which, apart from iframes, do not support overlapping screen estate. Where simultaneous execution of more

than one application is supported, both foreground and background applications shall be supported simultaneously.

Where simultaneous execution of more than one application is supported, applications shall be recorded within a hierarchy of applications. Each object representing an application possesses an interface that provides access to methods and attributes that are uniquely available to applications. For example, facilities to create and destroy applications can be accessed through such methods.

4.4.4 The application tree

Where simultaneous execution of more than one application is supported, applications are organised into a tree structure. Using the `createApplication()` method as defined in 7.2.3.3, applications can either be started as child nodes of the application or as a sibling of the application (i.e. added as an additional child of this application's parent). The root node of an application tree is created upon loading an initial application URI or by creating a sibling of an application tree's root node. An OITF may keep track of multiple application trees. Each of these individual application trees are connected to a hidden system root node maintained by the OITF that is not accessible by other applications.

Applications created while the DAE environment is running (e.g. as a result of an external notification) that are not created through `createApplication()` shall be created as children of the hidden system root node.

4.4.5 The application display model

4.4.5.1 General

Applications shall be displayed on the OITF in one of the application visualization modes as defined in 4.5.7.

The mode used shall be determined prior to initialisation of the DAE execution environment and shall persist until termination or re-initialization of the DAE execution environment. The means by which this mode is chosen is outside the scope of this document.

Each application has an associated DOM window object and a DOM Document object that represents the document that is currently loaded for that application. Even “windowless” applications that are never made visible have an associated DOM Window object.

4.4.5.2 Manipulating an application's DOM Window object

Standard DOM window methods are used to resize, scroll, position and access the application document (see 4.5.7). Many browsers restrict the size or location of windows; these restrictions shall NOT be enforced for windows associated with applications within the browser area. Any area of the display available to DAE applications may be used by any application. Thus, ‘widget’-style applications can create a small window that contains only the application without needing to be concerned with any minimum size restrictions enforced by browsers.

4.4.6 The security model

Each application has a set of permissions to perform various privileged operations within the OITF. The permissions that are granted to an application are defined by the intersection of three permission sets:

- 1) the permissions requested by the application, using the mechanism defined in Clause 10;
- 2) the permissions supported by the OITF; some permissions may not be supported due to capability restrictions (e.g. the *permission_pvr* permission will never be granted on a receiver that does not support PVR capability);

- 3) the permissions that may be granted, as determined by user settings or configuration settings specified by the operator (e.g. blacklists or whitelists; see Clause 10 for more information); this is a subset of 2), and may be different for different users.

4.4.7 Inheritance of permissions

Applications created by other applications (e.g. using the methods described in 5.1.2.2 or 5.1.2.4) shall not inherit the permissions issued to the parent application. The permissions granted to the new application will be defined by the mechanism specified in Clause 10.

When an application uses cross-document messaging (see the `window.postMessage()` method defined in HTML5 Web Messaging as referenced in IEC 62766-5-2 to communicate with another application, any action carried out in response to the message shall take place in the security context of the application to which the message was sent. Applications should take care to ensure that privileged actions are only taken in response to messages from an appropriate source.

4.4.8 Privileged application APIs

4.4.8.1 General

The privilege model implemented with applications is based upon requiring access to the Application object representing an application in order to access the privileged functionality related to application lifecycle management and inter-application communication.

4.4.8.2 Compromising the security

Since applications have access to Application objects, it is possible for applications to compromise the security of the framework by passing these objects to untrusted code. For example, an application could raise an event on an untrusted document and pass a reference to its Application object in the message. Where simultaneous execution of more than one application is supported, any calls to methods on an Application object from pages not running as part of an application from the same provider shall throw an error as defined in 10.1.2.

4.4.9 Active applications list

Where simultaneous execution of more than one application is supported, the OITF shall maintain a list of application nodes ordered in a "most recently activated" order – the active applications list. This list is used by the cross-application event dispatch algorithm as defined in 4.5.8 and is not directly visible to applications.

An application is activated through calling the `activateInput()` method of the application node. This marks an application as active and shall insert the application at the start of the active application list (removing it from the list first if it is already present).

An application is deactivated through the `deactivateInput()` method of the application node. This marks an application as inactive and shall remove it from the active application list.

The currently active application is the application at the start of the active application list.

This document does not define any behaviour if more than one copy of the browser is executing.

4.4.10 Widgets

DAE Widgets are a specialization of DAE applications and share aspects with W3C Widgets.

W3C Widgets are standardized by the “Widgets 1.0 family of specifications” as described in subclause 1.4 of W3C Widget Packaging and XML Configuration. Clause 11 specifies which parts of W3C Widgets specifications are supported by DAE Widgets. From here on, when using the word “Widget”, this means DAE Widgets as defined in this document.

Widgets can be primarily seen as packaged DAE applications. Since they are packaged, it is possible to have a single download and installation on an OITF. Widgets may also be installed on an OITF via non-HTTP distribution channels and even over off-network channels (e.g. a USB thumb drive). Packaging also provides an easy way to deploy and/or update applications on the OITF when it is installed in the home. The packaging and configuration of a DAE Widget is described in 11.2.

Since DAE Widgets are DAE Applications, everything that is defined for a DAE Application is also applicable to a Widget unless specified. Furthermore Widgets have several specific features as defined in Clause 11.

4.4.11 Origin for broadcast-delivered documents

For documents that are delivered by an object carousel in a broadcast channel (as defined in 4.1 of IEC 62766-2-1:2016, the following shall apply:

- for each broadcast channel, the OITF shall generate a unique origin as defined in IETF RFC 6454;
- these origins shall be of the form scheme/host/port tuple where the OITF does not use the same scheme for any documents other than those delivered by a broadcast object carousel;
- the origin shall be the one for the channel from which the document was loaded regardless of any subsequent channel changes.

Specifically, this origin shall be used with the Web Storage API as referenced in IEC 62766-5-2.

4.5 Resource management

4.5.1 General

Subclause 4.5 describes how resources (including non-granular resources such as memory and display area) are shared between multiple applications that may be running simultaneously. Applications should be able to tolerate the loss of scarce resources if they are needed by another application, and should follow current industry best practices in order to minimize the resources they consume.

This document is silent about the mechanism for sharing resources between DAE applications and other applications running on the OITF. In the remainder of this subclause and this document, the term application refers solely to DAE applications.

4.5.2 Application lifecycle issues

Where simultaneous execution of more than one application is supported, if an application attempts to start and not enough resources are available, the application with the lowest priority may be terminated until sufficient resources are available for the new application to execute or until no applications with a lower priority are running. Applications without a priority associated with them (e.g. applications started by the DRM agent, see 5.1.2.7) shall be assumed to have a priority of 0x7F.

Applications may register a listener for ApplicationUnloaded events (see 7.2.2.5) to receive notification of the termination of a child application, where simultaneous execution of more than one application is supported.

Failure to load an asset (e.g. an image file) or CSS file due to a lack of memory shall have no effect on the lifecycle of an application, but may result in visual artefacts (e.g. images not being displayed). Failure to load an HTML file due to a lack of memory may cause the application to be terminated.

4.5.3 Caching of application files

Application files may be cached on the receiver in order to improve performance; this document is silent about the use of any particular caching strategy.

4.5.4 Memory usage

Applications should use current industry best practices to avoid memory leaks and to free memory when it is no longer required. In particular, applications shall unregister all event listeners before termination, and should unregister them as soon as they are no longer required.

Where available, applications shall use explicit destructor functions to indicate to the platform that resources may be re-used by other applications.

Applications may use the `gc()` method on the `application/oipfApplicationManager` embedded object to provide hints to the OITF that a garbage collection cycle should be carried out. The OITF is not required to act on these hints.

The `LowMemory` event described in 7.2.2.5 shall be generated when the receiver is running low on memory. The amount of free memory that causes this event to be generated is implementation dependent. Applications may register a listener for these events in order to handle low-memory situations as they choose best.

4.5.5 Instantiating embedded objects and claiming scarce system resources

The objects defined in Clause 7 are embedded objects. These are typically instantiated through the standard DOM 2 methods for creating HTML objects or the `oipfObjectFactory` as defined in 7.1.

All embedded objects as defined in Clause 7 shall not claim scarce system resources (such as a hybrid tuner) at the time of instantiation. Hence, instantiation shall not fail if the object type is supported (and sufficient memory is available).

For each embedded object for which scarce resource conflicts may be a problem, the state diagram and the accompanying text define how to deal with claiming (and releasing) scarce system resources.

Once an OIPF embedded object has been instantiated, dynamic change of its MIME type, which could cause the properties and methods associated with the object to change, shall be ignored.

For instance, it is possible to change the MIME type of an A/V Control embedded object from `video/mpeg` to `video/mp4` but it is not possible to change the MIME type of an OIPF embedded object from `"application/oipfApplicationManager"` to `"application/oipfConfiguration"`.

4.5.6 Media control

If insufficient resources are available to present the media, the attempt to play the media shall fail. For the video/broadcast object, this shall be indicated by a `ChannelChangeError` event with a value of 11 for the error state. For an A/V Control object, the error property shall take the value 3.

When the video/broadcast or A/V Control object either is instantiated in the DYNAMIC_ALLOCATION model or transitions to the DYNAMIC_ALLOCATION model, scarce resources such as a media decoder shall only be claimed following a call to the `bindToCurrentChannel()`, `setChannel()`, `nextChannel()` or `prevChannel()` methods on a video/broadcast object or the `play()` method on an A/V Control object. By implication, instantiating a video/broadcast or A/V Control object does not cause the media referred to by the object's data attribute to start playing immediately. See 7.13.2.2 for details of when scarce resources are released by a video/broadcast object and 7.14.2.2 when scarce resources are released by an A/V Control object.

Scarce resources can be claimed by the video/broadcast or A/V Control object at instantiation time by specifying the `requiredCapabilities` parameter. In this case, the STATIC_ALLOCATION method is used and the scarce resources are held by the object until it is either destroyed or the `release()` method is called.

This part is intentionally silent about handling of resource use by embedded applications including scheduled recordings.

4.5.7 Use of the display

A compliant OITF shall support at least one of the following application visualization modes for managing the display of applications.

- a) Multiple applications may be visible simultaneously, with each application having a full-screen window, with the OITF managing the focus. Setting parts of an application to be transparent shall cause the following to be visible except where the application has drawn UI elements:
 - firstly any applications with a lower Z-index,
 - secondly video (if the hardware supports overlay as per the `<overlay*>` elements defined in 9.2 for the capability profiles).

In this mode, applications from the same service provider that are intended to run simultaneously should take care to coordinate their use of the display in order to ensure that important UI elements are not obscured.

- b) Multiple applications may be visible simultaneously, with the OITF managing the size, position, visibility and focus between applications.
- c) Only one application is visible at any time; switching to a different application either hides the currently visible application (where simultaneous execution of more than one application is supported) or terminates the currently visible application (where simultaneous execution of more than one application is not supported). The mechanism for switching between applications is implementation-dependent. In this case, the `show()`, `hide()`, `activateInput()` and `deactivateInput()` methods of the Application object provide hints to the execution environment about whether the user should be notified that an application requires attention. The mechanism for notifying the user is outside the scope of this document.

Applications shall be created with an associated DOM window object, that covers the display area made available by the OITF to a DAE application. The size of the DOM window can be retrieved through properties `innerwidth` and `innerHeight` of the DOM window object.

Any areas of the browser area outside the DOM window that become visible when it is resized shall be transparent – any video (if the hardware supports overlay as per the `<overlay*>` elements defined in 9.2 for the capability profiles) or applications (if multiple applications can be visible simultaneously) with a lower Z-index will be visible except where the application has drawn UI elements.

Broadcast-related and service provider related applications shall initially be created as invisible to avoid screen flicker during application start-up. Once loaded (as shall be indicated through an `onload` event handler), the application then typically calls the `show()` method of

its parent `Application` object. Broadcast-independent applications shall initially be created as visible and need not call these methods.

If the application does not ever need to be visible, then its DOM window object will never be shown. In that case, the application should take steps to avoid being formatted to reduce computation and memory overheads. This is typically accomplished by setting the default CSS style of the document's `BODY` element to `visibility:hidden`.

Because all applications have associated DOM window objects, it is possible to make any application visible even if it is not normally intended to be visible. This is of particular benefit during debugging of hidden service type applications.

Application developers should explicitly set the background colour of the application `<body>` and `<html>` elements.

Setting the background colour to 'transparent' (e.g. using the CSS construct `html, body { background-color: transparent; }`) will allow the underlying video to be shown for those areas of the screen that are not obscured by overlapping non-transparent (i.e. opaque) children of the `<body>` element.

Changing the visibility of an application by calling method `show()` or `hide()` on the `Application` object shall not affect its use of resources. The application still keeps running and listens to events unless the application gets deactivated (see 4.4.9) or destroyed (see 5.1.3).

4.5.8 Cross-application event handling

4.5.8.1 General

As defined in the DOM Level 3 Events specification as referenced in IEC 62766-5-2, standard DOM events are raised on a specific node within a single document. This document extends the event capability of the OITF through cross-application events handling, but does not change the DOM2 event model for dispatching events within documents. Where simultaneous execution of more than one application is supported, an OITF shall implement the cross-application events and cross-application event handling model described in this subclause.

An OITF shall implement the following cross-application event handling model. Cancelling the propagation of an event in any phase shall abort further raising of the event in subsequent phases: if an event is eligible for cross-application event handling (see below for more information) and is targeted at a node in the most recently activated application, then dispatch the event to that node using the standard DOM bubbling/capturing of events. Default actions normally taken by the browser upon receipt of an event shall be carried out at the end of this step, unless overridden using the existing DOM methods (i.e. using method `preventDefault()`).

If the cross-application event is not prevented from being propagated beyond the document root node of the application by using the existing DOM methods, the event is dispatched to other active applications in the application hierarchy using the active applications list described in 4.4.9. The OITF shall iterate over the applications in the active application list, from most recently activated to least recently activated, dispatching the event to the `Application` object of each application in turn. Note that the event shall NOT be dispatched to the document, and default browser action shall NOT be carried out during this phase. Cancelling the propagation of an event in this phase shall abort further raising of the event in subsequent applications.

Event listeners for cross-application events are registered and unregistered using the same mechanism as for DOM2 events. Listeners for cross-application events may be registered on the `Application` object as well as on nodes in the DOM tree.

The following events are valid instances of cross-application events and are applicable for cross application event handling, as shown in Table 1.

Table 1 – Events applicable for cross application event handling

System event	Description
KeyPress	Generated when a key has been pressed by the user. May also be generated when a key is held down during a key-repeat.
KeyUp	Generated when a key pressed by the user has been released.
KeyDown	Generated when a key has been pressed by the user.

The KeyPress, KeyUp and KeyDown events are all targeted cross-application events. The events are targeted at the node that has the input focus.

All events dispatched using the standard `dispatchEvent()` method are normal DOM events, not cross-application events. As defined in HTML5 Web Messaging as referenced in IEC 62766-5-2, the OITF shall support the `window.postMessage()` method for cross-document messaging. The method takes two arguments; a message (of type String) to be dispatched and the `targetOrigin`, which defines the expected origin (i.e. domain) of the target window, or "*" if the message can be sent to the target regardless of its origin. The target of the event is the "window" of a specific application. Applications can use this method to send events to other applications. The receiving application may receive those events and interpret them, or may dispatch them in its DOM using standard DOM `dispatchEvent()` methods.

The visibility of an application shall not affect the cross-application event handling algorithm as defined above – an active application shall receive cross-application events even when it is not visible.

Incoming key events are dispatched using the cross-application event handling algorithm as defined above.

NOTE This event dispatch model enables key events to be dispatched to multiple applications. Applications wishing to become the primary receiver for key events should call `Application.activateInput()`. Even though `Application.activateInput()` is called, another application may subsequently be activated. In order to ensure that sensitive key input (e.g. PINs or credit card details) is limited only to the application it is intended for, applications should check that they are the primary receiver of the key events (using the `Application.isPrimaryReceiver` property and/or the `ApplicationPrimaryReceiver` and `ApplicationNotPrimaryReceiver` events defined in 7.2.7) and should 'absorb' key events by calling the `stopPropagation()` method on the DOM2 key event.

4.5.8.2 Behaviour of the BACK key

OIPF applications may use the methods on the History object to navigate the history list. The history list shall not go back beyond the initial page of an OIPF application.

If a remote features a "back" or "back up" key, or one offering similar functionality, the OITF shall handle this key as described below:

A `VK_BACK` key event shall be dispatched to applications following the normal key handling process described in 4.5.8.

The default behaviour of the `VK_BACK` key event is implementation-dependent but the OITF shall not load the previous page in its history list for DAE applications.

4.5.9 Tuner resources

Tuners can be used for recording, scanning or watching broadcast channels (e.g. DVB-T). The priority relating to resource management is as follows. Recording has the highest priority,

viewing a channel has the lowest priority. A record request shall not be automatically interrupted by viewing a channel or a scan request. Note that to free the tuner for viewing requires interrupting the recording first.

4.6 Parental access control

A number of different approaches to parental access control is permitted.

a) Enforcement in the network.

An IPTV service provider may manage parental access control completely in the network. Applications running on application servers back in the network may decide to block access to content or arrange a DAE application to ask for a PIN code as necessary. This approach can apply to any kind of content – streaming on-demand content, IP broadcast content and to downloaded content.

No specific support is needed for this approach in the specification.

b) Enforcement in the OITF CSP / CSPG for protected MPEG-2 TS content

IPTV service providers may use the content protection mechanism for protected content to enforce access control to protected content. If used, this enforcement will happen in the OITF and, in some cases, in the CSP Gateway as well. In this approach, the content protection mechanism in the OITF would ask for PIN codes as needed.

The OITF CSP/CSPG-based enforcement of this approach and link to DAE API and events are defined in:

- 4.2.6.2 of IEC 62766-7:2017, for CSP terminal centric approach,
- 4.3.3 and 4.3.4.5.1.1.3 of IEC 62766-7:2017 for CI+ CSP Gateway centric approach
- 4.3.3 and 4.3.5.6.1 of IEC 62766-7 for DTCP-IP CSP Gateway centric approach

c) Enforcement in the OITF

An OITF may enforce parental access controls itself. Examples include embedded applications offering access to:

- IP delivered content based on information delivered to the metadata CG client,
- classical broadcast content in hybrid OITFs,
- content delivered to the OITF (either streaming or downloaded).

In approaches b) and c), PIN dialogs would be generated by code forming part of the OITF implementation. The APIs in 7.9 provide some control over these dialogs. The PIN would typically be configured by an embedded application but may also be configured by a DAE application using the optional APIs defined in 7.3.3.

These approaches b) and c) are reflected in a number of failure modes as defined in the following subclauses of this document:

- for broadcast channels (both IP and hybrid), in 7.13.2.3, see `onChannelChangeError` where `errorState 3` is defined as "parental lock on channel";
- parental rating errors and parental rating changes during playback of A/V content through the AV Control APIs defined in 7.14 and the `video/broadcast` object are reported according to the mechanism described in 7.14.6 and 7.13.6 respectively.

NOTE Due to the variation in regulatory requirements and deployment scenarios, this standard is intentionally silent about which of these approaches or combination of approaches is used.

4.7 Content download

4.7.1 General

The requirements in this subclause apply if the <download> element has been given value "true" in the OITF's capability profile as specified in 9.3.5.

4.7.2 Download manager

An OITF shall support a native download manager (i.e. "Content Download" component) to perform the actual download and storage of the content, and which allows the user to manage (e.g. suspend/resume, cancel) and monitor the download, in a consistent manner across different service providers. The download manager shall continue downloading as a background process even if the browser does not have an active session with the server that originated the download request anymore (e.g. has switched to another DAE application), even after a device power-down or network failure, until it succeeds or the user has given permission to terminate the download (see 4.7.5 on HTTP Range support to resume HTTP downloads after a power/network failure).

The native download manager shall be able to offer a visualization of its status through the application/oipfStatusview embedded object as defined in 7.15.2.1.

If the attribute manageDownloads of the <download> element in the client capability description is unequal to "none", the native download manager shall offer control over the active downloads through the JavaScript API defined by the application/oipfDownloadManager embedded object in 7.4.4.

NOTE 1 Once sufficient data of the content has been downloaded, the content may be played back using a native application, and may be played back using an A/V Control object. In the latter case, see method setSource() in 7.14.8 for more information.

NOTE 2 Annex D clarifies the content download usage scenario in more detail.

4.7.3 Content access download descriptor

An OITF shall support parsing and interpretation of the content access download descriptor document format with the specified semantics, syntax and MIME type as specified in Annex E.

4.7.4 Triggering a download

4.7.4.1 General

An OITF shall support a non-visual embedded object of type "application/oipfDownloadTrigger", with the JavaScript API as defined in 7.4.2 and 7.4.3 to trigger a download.

Subclauses 4.7.4.2 to 4.7.4.5 define some details about the different ways of triggering a download.

4.7.4.2 Using the registerDownload() method

The registerDownload() method takes a content access download descriptor as one of its arguments and passes it to the underlying native download manager in order to trigger a download. The following requirements apply.

The content access download descriptor may be created in JavaScript or may be fetched using XMLHttpRequest. To this end, the OITF shall pass the data inside the content access download descriptor into the XMLHttpRequest.responseXML property in JavaScript for further processing, if the OITF encounters an HTTP response message with the Content-Type of "application/vnd.oipf.ContentAccessDownload+xml", as the result of an XMLHttpRequest.

NOTE The behaviour in other cases when the OITF encounters an HTTP response message with the Content-Type "application/vnd.oipf.ContentAccessDownload+xml", for example whilst following a link as specified by an anchor element (<a>), is not specified in this document.

If the OITF supports a DRM agent with a matching DRMSystemID as per 9.3.11, the OITF shall pass included DRM-information as part of the <DRMControlInformation> elements of a content-access download descriptor to the DRM agent.

If the content access descriptor contains multiple content items to be downloaded, then all items are considered to belong together. Therefore, the download of each individual content item has the same download identifier in that case (whereby the ContentID may be used for differentiation). The order by which the items are downloaded is defined by the OITF.

4.7.4.3 Using the registerDownloadURL() method

The registerDownloadURL() method takes a URL as one of the arguments and passes it to the underlying native download manager in order to trigger a download. The URL may point to any type of content. The URL may also point to a content access download descriptor (i.e. with argument contentType having value "application/vnd.oipf.ContentAccessDownload+xml"). In that case, the method returns a download identifier. The OITF will then fetch the content access download descriptor, after which the same shall happen as if method registerDownload() as defined in 4.7.4.2, with the given content access download descriptor as argument was called.

4.7.4.4 Using the optional registerDownloadFromCRID() method

The registerDownloadFromCRID() method is an optional method as defined in 7.4.3 and takes a CRID as one of its arguments that is passed to the underlying native download manager in order to trigger a download.

4.7.4.5 General behaviour regarding triggering a download

The following are general behavioural requirements apply to triggering downloads.

Fetching the content will typically be initiated immediately. However, the OITF may defer the download to a later time.

An OITF should offer an easy way to continue the UI interaction with the server from which a download has been initiated, e.g. allowing him/her to continue browsing on the page that triggered the download.

An OITF should inform the user if the content-type of a content item being retrieved cannot be interpreted by the OITF.

4.7.5 Download protocol(s)

The OITF shall support the HTTP protocol for download as specified in 6.3.4 of IEC 62766-4-1:2017. In addition, the OITF shall support the following requirements.

As specified in 6.3.4 of IEC 62766-4-1, if a server offers a content item for download using HTTP, the server shall make sure that HTTP Range requests as defined in IETF RFC 2616 are supported for HTTP GET requests to the URI of that downloadable content item, in order to be able to resume downloads (e.g. after power or network failure).

If the OITF receives an HTTP 404 "File Not Found" status code, the OITF shall stop its attempts to resume the download, and go to a "Failed Download" state. The handling of other error codes is implementation dependent.

If, after downloading, a content item the size of the downloaded content item does not match the indicated size parameter, or the value for the optional attribute “MD5Hash” of the given <ContentURL> does not match the hash of the downloaded content, the OITF should remove the downloaded content item.

Integration with download protocols other than HTTP are not specified in this document.

4.8 Streaming CoD

4.8.1 General

Subclause 4.8 defines the content-on-demand streaming interfaces for both DRM-protected and non-DRM protected content.

4.8.2 Unicast streaming

4.8.2.1 General

There are three mechanisms by which a reference to content can be passed from a DAE application to the OITF.

- By setting the data property of a A/V Control object as defined in 7.14 to the reference. The application shall set the type attribute to the MIME type of the content referred to by the value of the data attribute to provide a hint about the expected content type, in order for the browser to instantiate the proper object to play the content.
- By setting the src attribute of a <video> element to the reference.
- By including the reference in the <ContentURL> element of a Content Access Streaming descriptor as defined in 7.14.3 and then setting the data property of an A/V Control object as defined in 7.14 to be a reference to that Content Access Streaming Descriptor. In this case the application shall set the type attribute to "application/vnd.oipf.ContentAccessStreaming+xml".

EXAMPLE `<object id="d1" data=http://www.openiptv.org/fetch?contentID=25 type="application/vnd.oipf.ContentAccessStreaming+xml" width="200" height="100"/>`

There are five different possible formats for a reference to unicast streaming content:

- a Public Service Identifier (PSI) as defined in Protocol Specification IEC 62766-4-1;
- an HTTP URL directly referencing the content to be streamed;
- an RTSP URL directly referencing the content to be streamed;
- an HTTP or HTTPS URL referencing a HAS MPD;
- an HTTP or HTTPS URL referencing a MPEG DASH MPD.

All of the mechanisms that an OITF supports shall be supported with all formats of a reference that an OITF supports.

4.8.2.2 HTTP Adaptive Streaming

If the OITF supports HAS content, then it shall support the MIME type as specified for the Media Presentation Descriptor (MPD) in 3GPPTS 26.234, i.e. "video/vnd.3gpp.mpd", and in the HAS specification IEC 62766-2-2. If the OITF supports MPEG DASH content, then it shall support the MIME type as specified for the Media Presentation Descriptor (MPD) in Annex C of ISO/IEC 23009-1:2014, i.e. "application/dash+xml".

The MPD shall be retrieved by specifying a URL. To this end, the OITF shall fetch the MPD from the URL, after which the MPD shall be interpreted and an initial (set of partial) representation(s) selected.

When the URL is passed to the OITF in the data property of a A/V Control object as defined in 7.14, and either a HAS MPD is not valid according to the XML Schema and semantics as defined in Annex A of IEC 62766-2-2:2016 or an MPEG DASH MPD is not valid according to the XML Schema and semantics as defined in ISO/IEC 23009-1 or IEC 62766-2-2, then the A/V Control object shall go to play state 6 ('error'), with error value 4 ('content corrupt or invalid').

If the OITF supports HAS content, then HAS shall also be supported through the video/broadcast object for live content. If the OITF supports MPEG DASH content, then MPEG DASH shall also be supported through the video/broadcast object for live content. This shall be done using Channel objects returned from calls to the createChannelObject(Integer idType, Integer onid, Integer tsid, Integer sid, Integer sourceID, String ipBroadcastID) method where the idType argument is ID_IPTV_URI and the ipBroadcastID argument is a URL which points to an MPD for Scheduled Content (live streaming) over HTTP.

4.8.2.3 Multicast streaming

If an OITF has indicated support for IPTV channels through a <video_broadcast> element with type ID_IPTV_* (as defined in 7.13.12.2), the OITF shall support passing a content-access descriptor through the 'contentAccessDescriptorURL' argument of the 'setChannel'-method of the video/broadcast' object (as defined in 7.13.2.4). If the content-access descriptor includes DRM information, the OITF shall pass this information to the DRM agent.

4.9 Scheduled content

4.9.1 General

If an OITF has indicated support for playback and control of scheduled content through the <video_broadcast> element, then it shall support the "video/broadcast" embedded object defined in 7.13.2. In addition, it shall adhere to the requirements for conveyance of the channel list as specified in 4.9.2. To protect against unauthorized access to the tuner functionality and people's personal favourite lists, the OITF shall adhere to the security model requirements as specified in 10.1, in particular the tuner-related security requirements in 10.1.4.2.

NOTE 1 This subclause and subclause 7.13 are focused on control and display of scheduled content received over local tuner functionality available to an OITF. The term "tuner" is used here to identify a piece of functionality to enable switching between different types of scheduled content services that are identified through logical channels. This includes IP broadcast channels (using the mechanisms for Scheduled Content defined in IEC 62766-4-1), as well as traditional broadcast channels received over a hybrid tuner.

NOTE 2 The APIs in this subclause allow for deployments whereby the channel line-up and favourite lists for broadcasted content are managed by the client, the server, or a mixture thereof.

4.9.2 Conveyance of channel list

4.9.2.1 General

To enable a service to control the tuner functionality on an OITF, the OITF needs to convey the channel list information that is managed by native code on the OITF device to the service (either the channel list information is provided locally on the OITF via JavaScript, or the channel list is communicated directly to a server). This information includes the list of uniquely identifiable channels that can be received by the physical tuner of a hybrid device, including information about how the channels are ordered and whether or not these channels are part of zero or more favourite lists. It also includes the channel line-up and the favourite lists that may be managed by an OITF for IP broadcast channels.

The API supports two methods of conveying the channel list information to a service:

- a) method 1: through JavaScript, by using the method "getChannelConfig()", as defined in 4.9.2.2;

- b) method 2: through an HTTP POST message that is sent upon the first connection to a service that requires tuner control, as defined in 4.9.2.3.

An OITF shall support method 1, and should support method 2.

If an OITF conveys the channel list information using the HTTP POST message defined in method 2, then the server shall, if it supports method 2, receive the conveyed channel list information and should rely on this information for the purpose of exerting tuner control. If a service supports using the channel list information sent through the HTTP POST method to exert tuner control, the server shall indicate this compatibility with method 2 using the `postList` attribute specified in 9.3.2 (i.e., `<video_broadcast postList="true">true</video_broadcast>`) in the server capability description.

If the server does not support method 2, the service shall rely on the `getChannelConfig()` method defined in 7.13.2.4 to access the channel list information. If an OITF does not support method 2, the HTTP message of the first connection to the service that requires tuner control shall be an HTTP GET message with an empty payload and the service shall instead rely on the `getChannelConfig()` method defined in 7.13.2.4 to access the channel list information. If support for method 2 is indicated by both the OITF and the server (through respective capability exchanges), the OITF shall convey the channel list information using method 2.

If an OITF does not manage/maintain the channel line-up (i.e. does not have a locally stored channel line-up), the `getChannelConfig()` method described in 7.13.2.4 shall return `null`, and the HTTP message described in 4.9.2.3 shall be an HTTP GET message with an empty payload. In that case, the application may use the `createChannelObject()` method as defined in 7.13.2.4 to create channel objects that can be used on subsequent `setChannel()` requests, and in this way can manage/maintain its own channel list.

NOTE Conveyance of the channel list shall adhere to the security model requirements as specified in 10.1.4.2 and 10.1.4.2.2.

4.9.2.2 Method 1: JavaScript method "getChannelConfig()"

The OITF shall support method "getChannelConfig()" as defined in 7.13.2.4 for the video/broadcast embedded object. This method returns a `ChannelConfig` object as defined in 7.13.10.

4.9.2.3 Method 2: HTTP POST message

If an OITF supports sending the channel list through HTTP POST and a server has indicated that it uses the posted channel list information to exert control of the tuner functionality of an OITF (i.e. using attribute `postList="true"` in the server capability description) for a particular service, then the OITF shall issue an HTTP POST over TLS if it decides to connect to that service. The body of the HTTP POST over TLS request shall contain the client channel listing, which shall adhere to the semantics, syntax and XML Schema that are defined for the client channel listing in Annex G. The server shall silently ignore unknown elements and attributes that are part of the client channel listing.

The server shall return a HTML document.

If the favourite lists are not (partially) managed by the OITF, the client channel listing shall neither contain the "FavouriteLists" nor the "CurrentFavouriteList" element.

4.9.3 Conveyance of channel list and list of scheduled recordings

This subclause shall apply to OITFs that have indicated `<recording>true</recording>` as defined in 9.3.4 in their capability profile.

To enable a service to schedule recordings of content that is to be broadcast on specific channels, the OITF needs to convey the channel list information that is managed by the native

code on the OITF. This information typically includes the channel line-up of the tuner of a hybrid device. The conveyance of channel list information and scheduled recordings is based on the same two methods of conveying the channel list information to a service as defined in 4.9.2:

- 1) method 1: through JavaScript, by using the method "getChannelConfig()". To this end, the OITF shall support method "getChannelConfig()" as defined in 7.10.2.2 for the application/oipfRecordingScheduler object.
- 2) method 2: through an HTTP POST message as defined in 4.9.2.3 that is sent upon the first connection to a service that has indicated that it requires control of the recording functionality and that has indicated compatibility with method 2 using the postList attribute specified in 9.3.4 (i.e., <recording postList="true">true</recording>), in the server capability description for a particular service.

An OITF shall support method 1, and should support method 2. If support for method 2 is indicated by both the OITF and the server (through respective capability exchanges), the OITF shall convey the channel list information using method 2. Otherwise, the HTTP message of the first connection to the service that requires tuner control shall be an HTTP GET message with an empty payload.

If a server has indicated that it requires control of both the tuner functionality and the recording functionality available to an OITF (i.e. by including both <video_broadcast> and <recording> with value true in the OITF's capability description), the body of the HTTP POST message shall contain a single instance of the client channel listing whereby the <Recordable> element defined in Annex G shall be used to indicate whether channels that can be received by the tuner of the OITF can be recorded or not.

If an OITF does not manage the channel line-up, the getChannelConfig() method described in 7.10.2.2 shall return null, and the HTTP message described in 4.9.2.3 shall be an HTTP GET message with an empty payload.

In addition, the OITF shall also support method getScheduledRecordings() as defined in 7.10.2.2. This method returns a ScheduledRecordingCollection object, which is defined in 7.10.4.

Note that the conveyance of the channel listing and the scheduled recordings is subject to the security model requirements specified in 10.1, and in particular the recording-related security requirements in 10.1.4.3.

4.10 DLNA RUI remote control function

4.10.1 General

Subclause 4.10 describes the DLNA RUI RCF (remote control function) and the interactions between the different entities involved. It builds on the RUI feature defined by the IEC 62481 series, the DLNA Networked Device Interoperability Guidelines (August 2009), and shows how the DLNA RUI can be integrated into an OITF and used by DAE applications.

The DLNA RUI RCF is the feature that enables a remote control device to be able to control the OITF or a DAE application running on it, from that remote control device. To support this feature, a remote control device shall support the DLNA RUIC function and an OITF shall support the DLNA RUIS function (as defined in 7.17).

The DLNA RUI RCF provides two main features:

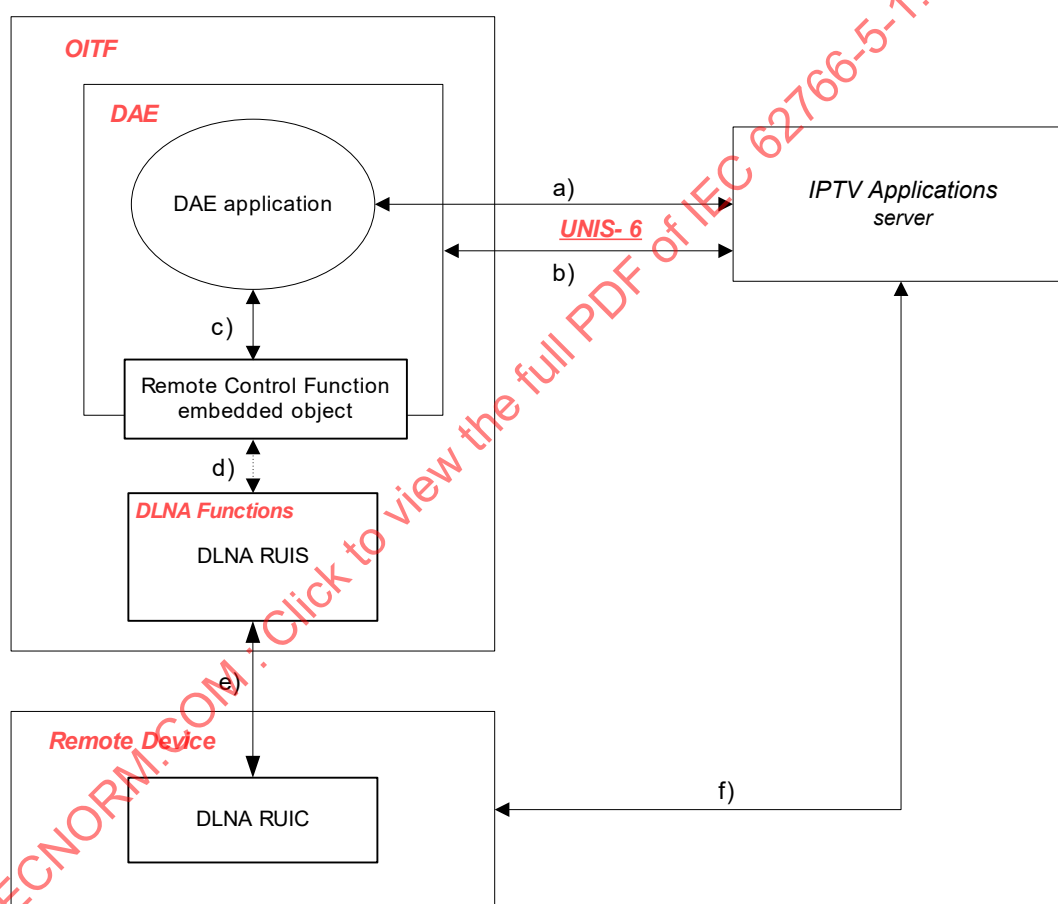
- providing a Control UI to the remote control device:
 - the Control UI is a CE-HTML document through which the user will control the OITF directly or a DAE application on the OITF. There are two options based on the origin of the Control UI for sourcing it as follows:

- sourcing the Control UI from the OITF itself;
- sourcing the Control UI from an IPTV Applications server via the OITF.
- interactions to exchange control messages and results:
 - the Control UI in the DLNA RUIC sends control messages to the OITF or DAE application and receives the corresponding results.

Subclause 4.10.2 introduces the interfaces between the entities that support the DLNA RUI RCF.

4.10.2 Interfaces used by the DLNA RUI remote control function

This subclause describes interfaces related to the DLNA RUI RCF. There are three entities (remote control device, OITF and IPTV applications server) that communicate with each other through the interfaces described in Figure 2.



Please see 4.10.2 for a description of the interfaces a) to f).

IEC

Figure 2 – OIPF architecture with DLNA RUI RCF scenario

Figure 2 shows the entities in the OIPF architecture involved in the DLNA RCF and the interfaces between them.

The dotted line "d)" between the RCF embedded object and the DLNA RUIS indicates that it is a local interface and hence not defined by this document. The detailed behaviour of each interface is defined as follows:

Interface a)

This interface is used to retrieve a control UI from an IPTV applications server by using XMLHttpRequest object (the Control UI retrieved through interface a) will be delivered to DLNA RUIC via interfaces c), d) and e), sequentially).

Interface b)

This interface is used by the DAE browser to retrieve a DAE application containing an RCF object when the DLNA RUIC requests a DAE application to execute in the OITF.

Interface c)

The DLNA RUI RCF APIs use this interface to enable a DAE application to get the request originating from the DLNA RUIC through an event dispatched by the OITF, and send the corresponding response or any other information to the DLNA RUIC via the DLNA RUIS.

Interface d)

This is a local interface that is used to pass messages between an RCF object in a DAE application and the DLNA RUIS.

Interface e)

This is a DLNA RUI compatible interface which provides device discovery, sending/receiving HTTP messages and notifications.

When the DLNA RUIC is activated by a user, the DLNA RUIC searches for a DLNA RUIS and does a capability exchange. Then, the DLNA RUIC retrieves the XML UI Listing from the DLNA RUIS and displays it to the user. When the user chooses one of the Control UIs, the DLNA RUIC retrieves the selected Control UI from the DLNA RUIS in the OITF.

The Control UI may send an HTTP request to deliver a message (for example, plays an AV content) and receive a response from the DLNA RUIS.

This interface is also used for the DLNA RUIS to send a 3rd party notification defined in 5.6.1 of CEA-2014-A.

Interface f)

This interface is used by the selected Control UI (CE-HTML document) to retrieve resources (for example, images, CE-HTML documents, or CSS or JavaScript files) directly from the IPTV Applications server.

4.11 Power consumption

4.11.1 General

The power states described in this subclause relate to states exposed to the DAE application. There may be other states supported by the OITF, which are not described here.

The OITF will be in one of a number of power states. Its default state is "off", which consumes no power. The OITF shall support an "on" state where it is running in normal operation. The OITF shall support at least one standby state where nothing is being output to the display but power is consumed. An OITF may support two different standby states, "active standby" and "passive standby". An OITF in the "passive standby" state has the smallest possible power consumption (for example, average under 1W), which may be in line with European Commission Code of Conduct, US Energy Star or other regional requirements. In this state, the IR listener and wakeup clock may be active but no DAE application is active. The IR

listener allows the user to turn on the OITF using a remote control. A DAE application may use the wakeup clock to schedule the OITF enter the "active standby" state, for example to perform a recording.

Note there may be different levels of "active standby" state, but the assumption is that, at least, nothing is being output to the display and one or more DAE applications may execute in the background.

The following explanation describes the behaviour of the OITF when transitioning between the mentioned states and how a DAE application is affected.

A DAE application shall be able to execute in the "on" and "active standby" states but shall not be able to execute in the "off" or "passive standby" states.

When an OITF is turned "on" from an "off" state, a DAE application has to be explicitly selected by the user to be executed or the OITF has identified a DAE application to be auto-started. A DAE application has no direct control if it shall auto-start or not and this is left for the OITF to manage. A DAE application may auto-start if the Service Discovery and Selection has taken place and the user has selected a service provider.

When an OITF changes to an "off" or "passive standby" state from an "on" or "active standby" state, the DAE application shall get an `ApplicationDestroyRequest` event. The DAE application has an opportunity to take a final action and gracefully quit or it shall be killed forcibly.

4.11.2 DAE application wake-up support

4.11.2.1 General

The OITF may support wake-up requests from a "passive standby" state. There are two types of wake-up requests, one on an individual DAE application and one on the OITF. The supported wakeup is indicated in the power consumption capability information.

4.11.2.2 Single DAE application wakeup

The OITF may support wake-up requests for individual DAE applications when in "passive standby". Similar to a scheduled recording, a DAE application may need to execute at a predetermined time. At the wake-up point, the DAE application executes, and when it completes its task returns to a "passive standby" state by exiting.

There shall only be one wake-up request per DAE application. There may be multiple wake-up requests from different DAE applications which shall execute independently. The OITF shall silently ignore all wake-up requests whose timers expire when it is not in the "passive standby" state.

When the DAE application terminates and the OITF changes to an "active standby" or "on" state for other reasons than a wake-up request, the OITF shall not change power states.

Through capability information it is possible to determine if wake-up and standby modes are supported by OITF.

This is an example of how a DAE application may setup a wake-up request in OITF.

Precondition: the DAE application is actively running and the OITF is either in "on" or "active standby" states.

End user selects to go into "passive standby" natively.

An `ApplicationDestroyRequest` event is generated

The DAE application calls the `preparewakeupApplication()` method and sets a token, time for wake-up and URI associated with the DAE application. The DAE application then quits, e.g. by calling `destroyApplication()` on its parent `Application` object.

The OITF goes into "passive standby" state.

When the wake-up time triggers, the OITF changes to "active standby" and the DAE application is initiated with the URI specified in the prior call to `preparewakeupApplication()`.

The DAE application then runs `clearwakeupToken()` to get the token set in the prior call to `preparewakeupApplication()`.

The DAE application executes.

Once the DAE application completes execution, it shall exit. The OITF changes automatically to a "passive standby" state.

If the OITF is turned "on" while in this mode, the OITF shall not enter "passive standby" state.

4.11.2.3 OITF wakeup

The OITF may support wake-up requests for the OITF when in "passive standby". The application when receiving an event `onApplicationRequest` may request to wake-up the OITF at a set time using method `peparewakeupOITF()`.

OITF shall silently ignore all wake-up requests whose timers expire when it is not in the "passive standby" state.

4.11.3 OITF hibernate support

The OITF may support a hibernate mode that allows DAE applications and their state to be stored in memory when in a "passive standby" state. The support of a hibernate mode greatly reduces the start-up time for DAE applications (for example, start-up times of 3 s may be reached).

When the OITF resumes from the hibernate mode, it shall restore all of the previous DAE applications with their previous states and should assign the same resources to the DAE applications as they had prior to the hibernate mode. If this is not possible, the regular callback functions shall be used to inform the affected DAE application.

If hibernate mode is supported, the event `ApplicationHibernateRequest` is generated instead of `ApplicationDestroyRequest` when the OITF enters a "passive standby" state.

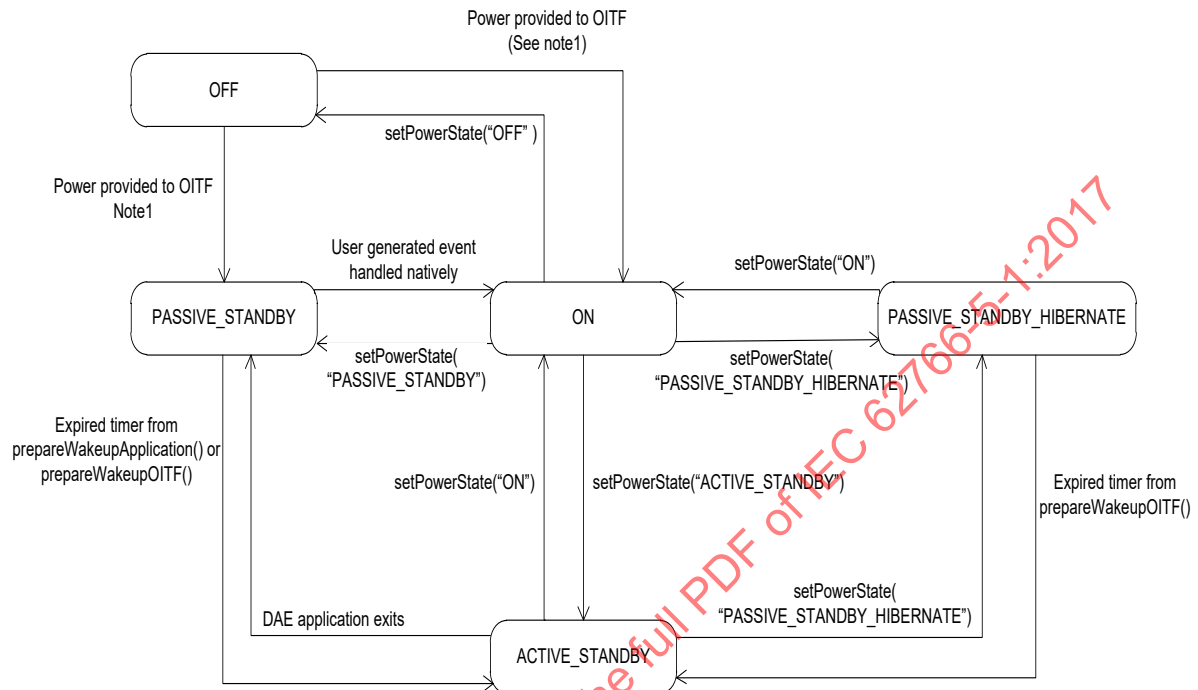
If the OITF supports hibernate mode only, the OITF wake-up request is supported. The single DAE application wake-up shall not be supported. The reason for this limitation is due to the difficulty to support both options.

A wake-up support shall not make the OITF resume from the hibernate mode. The wake-up support shall be supported independently.

The OITF shall indicate support for hibernate mode through the `<hibernateMode>` capability defined in 9.3.20.

4.11.4 State diagram for the power state

The state machine in Figure 3 provides an overview of the power state changes that may occur relating to power consumption. The transitions in the state machine due to `setPowerState()` may be also be triggered by user-generated events handled natively by the OITF.



IEC

NOTE The transition from the OFF state to the PASSIVE_STANDBY or ON states is manufacturer-dependent.

Figure 3 – State diagram of OITF power states

4.12 Display model

Annex H describes the logical display model of an OITF and the relationship between DAE application graphics and video.

5 DAE application model

5.1 Application lifecycle

5.1.1 General

Subclause 5.1 describes the lifecycle of a DAE application, including when an application is launched, when it is terminated and the behaviour when a DAE leaves the boundary of one application and enters another.

APIs related to DAE applications are described in 7.2.

5.1.2 Creating a new application

5.1.2.1 General

The present list defines a number of different application lifecycle models. These include:

- applications started through an OITF-specific user interface;

- using the `Application.createApplication()` API call;
- CE-HTML third party notifications;
- service provider related applications (from SD&S signalling);
- applications started by the DRM agent;
- applications provided by the AG through the remote UI;
- broadcast-related applications (either be from SD&S signalling or from broadcast signalling in a hybrid device);
- broadcast-independent applications;
- widgets.

5.1.2.2 Broadcast-independent applications

Broadcast-independent applications are started by fetching the first page of the application from a URL.

5.1.2.3 Applications started through an OITF-specific user interface

These shall be presented as broadcast-independent applications.

5.1.2.4 Using the `Application.createApplication()` method

Creating a new application is accomplished by creating a new `Application` object via the `Application.createApplication()` method. Calling this method will create a new application and add it to the application tree in the appropriate location.

```
// Assumes that the application/oipfApplicationManager object has the ID
// "applicationmanager"
var appMgr = document.getElementById("applicationmanager");
var self = appMgr.getOwnerApplication(window.document);

// create the application as a child of the current application
var child = self.createApplication( url_of_application, true );
```

The URL passed to the `createApplication()` method shall be one of the following:

- an HTTP or HTTPS URL referring to an XHTML page as defined by 6.1;
- an HTTP or HTTPS URL referring to an XML AIT as defined by 5.2.7.1;
- the DVB URI for launching service provider related applications signalled through SD&S as defined in 8.3;
- the DVB URI for launching broadcast-related applications from the current service signalled through SD&S as defined in 8.3. Where an OITF supports the MPEG-2 encoding of the AIT as defined in 5.2.7.2, this form of the DVB URI shall also be supported for launching broadcast-related applications from the current service when that service includes an MPEG-2 AIT.

5.1.2.5 CE-HTML third party notifications

The lifecycle of these is defined in CEA-2014-A and summarised in 5.3.2.

5.1.2.6 Starting applications from SD&S Signalling

These are described in 5.2. All applications started by SD&S signalling are treated as siblings and are children of the hidden system root node (see 4.4.4).

5.1.2.7 Applications started by the DRM agent

These shall be considered as broadcast-independent applications.

5.1.2.8 Applications provided by the AG through the remote UI

OITFs may include the capability to start these applications from an embedded application. OITFs shall include the ability for applications to discover these as defined by the "application/oipfGatewayInfo" embedded object in 7.7.2.

5.1.3 Stopping an application

The `destroyApplication()` method (as specified in 7.2.3.3) shall terminate the application. An application may register a listener on the `ApplicationDestroyRequest` event in order to perform any clean-up before being destroyed completely. After the `destroyApplication()` method returns, further execution of the specified application shall not occur.

When an application is terminated, all associated resources shall be freed (or marked available for garbage collection). Any active network-related sessions will be terminated. Any media content being presented by the application is stopped, although recordings or content downloads initiated by the application will not be affected.

NOTE Terminating an application does not imply any effect on the state of the DAE environment.

Additional requirements are defined for stopping selected service provider applications and applications that are part of scheduled content services in 5.2.4.3 and 5.2.3.2 respectively.

5.1.4 Application boundaries

All of the pages that make up an application are contained within its application boundary. This is the "fully qualified domain name" (FQDN) of the initial page of the application in the absence of a `simple_application_boundary_descriptor`.

If an `applicationBoundary` element is present in the SD&S signalling for an application as defined in ETSI TS 102 809, the application boundary shall also include the FQDNs listed in the `applicationBoundary` element. If this element is not present, then the application boundary shall consist of the FQDN of the initial page of the application.

For files requested with `XMLHttpRequest`, the same origin policy shall be extended using the application domain; i.e. any domain in the application domain shall be considered of same origin.

The OITF shall remove any IP address in the application boundary which is within the private address space as defined in IETF RFC 1918, before launching the application.

Extending the origin of `XMLHttpRequest` is potentially dangerous, and may lead to undesired leaking of private information. To make sure that the integrity of the user is not compromised, the OITF should include a mechanism which allows the user to exclude domains from application boundaries of applications.

5.2 Application announcement and signalling

5.2.1 Overview

This document defines 3 basic types of application:

- applications related to one or more broadcast TV or radio channels. These may run while one of the channels which they are related to is being presented by the OITF. These are signalled through the SD&S broadcast or package discovery records or included in an application discovery record which is referenced from the broadcast or package discovery record.
- applications related to the service provider selected through the service selection process. These may run at any time until the service provider selection process is repeated and a

different service provider selected. These are signalled through the SD&S service provider discovery record or included in an application discovery record which is referenced from the service provider discovery record.

- applications independent of either of the above. These may run at any time. These are started by other applications and are not signalled anywhere.

Each of these types is described in more detail in 5.2.2 to 5.2.8.

5.2.2 General

Subclause 4.4.4 describes how one application may start another application either as a sibling or as a child. All applications started via SD&S signalling as described in this subclause shall be started as children of the hidden system root node, as described in 5.1.2.6.

Any application may be signalled as AUTOSTART or PRESENT (see Table 3 and 5.2.4.3 of ETSI TS 102 809). Applications signalled as AUTOSTART are intended to be automatically started by the OITF. Applications signalled as PRESENT are intended to be started only by other applications. Broadcast-related applications may alternatively be signalled as KILL or PREFETCH.

It is up to the OITF manufacturer to ensure a good quality of experience concerning:

- navigation within a DAE application;
- accessing the available DAE applications, both available for launch, and those already running;
- managing the life cycles of all DAE applications able to be used concurrently.

It is outside the scope of this document whether there are dedicated keys on a remote control (e.g. the "menu", "home" or "guide" key), there is an entry in an on-screen menu or there are some other mechanism.

It is optional for the OITF to support an exit mechanism directly accessible by the end-user. If one is supported, it is outside the scope of this document whether this mechanism is a button on a remote control, an item in an on-screen menu or something else. If such a mechanism is supported then it shall only stop the application the end-user is currently interacting with and any child applications of that application. The parent application and any siblings shall not be stopped.

Additionally, any application may be stopped under the following circumstances:

- the application itself exits;
- its parent application exits;
- it is stopped by the application which started it or another application which has a reference to its application object;
- in response to changes in the application signalling as defined below for broadcast-related applications and service provider related applications.

In all these above cases, except the first (when an application itself exits), when an application is stopped by the OITF, an ApplicationDestroyRequest event (as defined in 7.2.7) shall be raised on the application. In the following error conditions, an application being stopped should have an ApplicationDestroyRequest event raised if this is possible:

- the OITF runs out of resources for applications and has to stop some of them in order to keep operating correctly;
- the OITF has determined that an application is non-responsive or has crashed.

5.2.3 Broadcast-related applications

5.2.3.1 General

Providers of broadcast TV channels may signal broadcast-related applications as part of the SD&S broadcast discovery record (see 4.3.3 of IEC 62766-3:2016, as well as 4.2.1 and 5.4.3.2 of ETSI TS 102 809). As an optimisation, broadcast-related applications that are associated with a group of channels may be signalled as part of the SD&S package discovery record (see 4.3.3 of IEC 62766-3:2016, as well as subclause 5.4.3.1 of ETSI TS 102 809). Broadcast-related applications may be included in the SD&S broadcast discovery or package discovery records or included in an application discovery record that is referenced from the broadcast discovery record.

Broadcast-related applications can also be signalled in-line in an MPEG-2 transport stream using the MPEG-2 encoding of the AIT as defined in 5.2.7.2 below.

When a broadcast TV channel starts being presented, the OITF shall follow the procedure defined in 5.2.3.3 below.

While a broadcast TV channel is being presented, the OITF shall monitor for changes in the SD&S information as defined by 5.2.2.3 of IEC 62766-3:2016.

When changes are detected, the OITF shall follow the procedure defined in 5.2.3.4.

NOTE The typical "red button" behaviour can be achieved by having the first page of an AUTOSTART broadcast related application be full screen and transparent to video except for an image showing a red button. Only when the user generates a "red" key event does the application display more of its user interface.

OITFs may include the capability to start and stop a broadcast-related DAE application instead of analogue teletext services as part of a scheduled content service or channel. Typically, this would re-purpose the same mechanism used to start an analogue teletext service – for example a "text" button on a remote control. These are identified using the application usage mechanism defined in ETSI TS 102 809 and 5.2.7.

5.2.3.2 Stopping

In addition to what is stated in 5.2.2, broadcast related applications are stopped when:

- changing between channels as defined in the procedure in 5.2.3.3;
- the OITF detects an update to the signalling for a currently presented channel as defined in the procedure in 5.2.3.4;
- the OITF stops presenting any broadcast channel.

5.2.3.3 Procedure for starting and stopping broadcast related applications on channel change

When a scheduled content service is selected, the following shall apply:

- the OITF shall determine if there are any applications signalled as part of the service as defined by 4.3.4.2 and 4.3.4.3 of IEC 62766-3:2016;
- applications that are related to that scheduled content service and which are signalled with a control code of AUTOSTART shall be started if not still running from any previously presented linear TV service; they shall be started commencing with the highest priority application working downwards in priority while resources in the OITF permit;
- applications that are related to that scheduled content service, which are signalled with a control code of AUTOSTART and that are already running from a previously presented scheduled content service shall:
 - continue to run uninterrupted if the serviceBound element of the ApplicationDescriptor in their signalling has value false;

- be stopped and re-started if the serviceBound element of the ApplicationDescriptor in their signalling has value true.
- applications that are related to that scheduled content service and which are signalled with a control code of PRESENT shall continue to run if already running but shall not be started if not already running;
- running applications from any previously presented scheduled content service that are not part of the new scheduled content service shall be stopped as part of the change of presented service.

Figure 4 shows the behaviour that shall apply when the selected channel changes.

IECNORM.COM : Click to view the full PDF of IEC 62766-5-1:2017

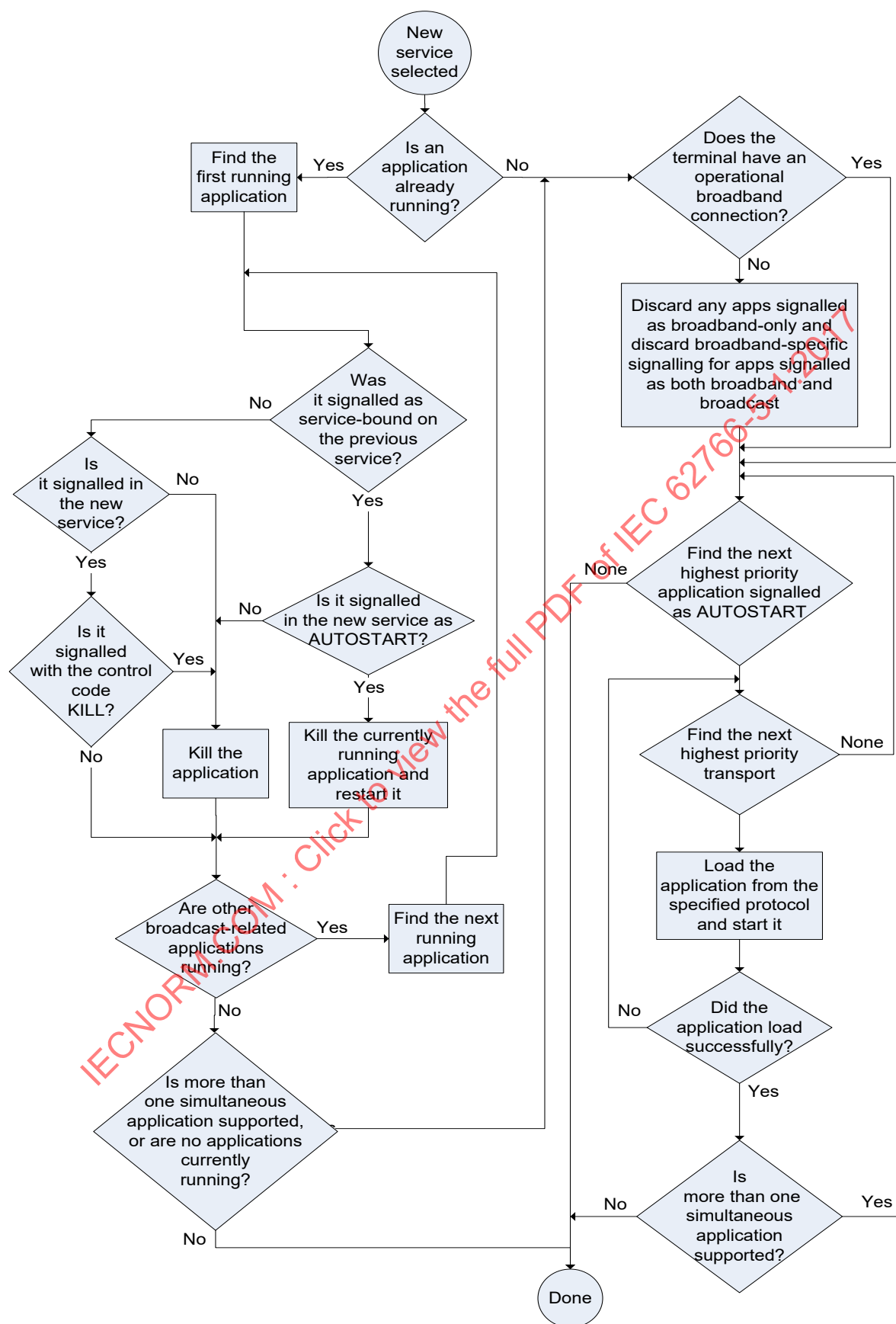


Figure 4 – Behaviour when the selected channel changes

5.2.3.4 Procedure for starting and stopping broadcast related applications when signalling is updated

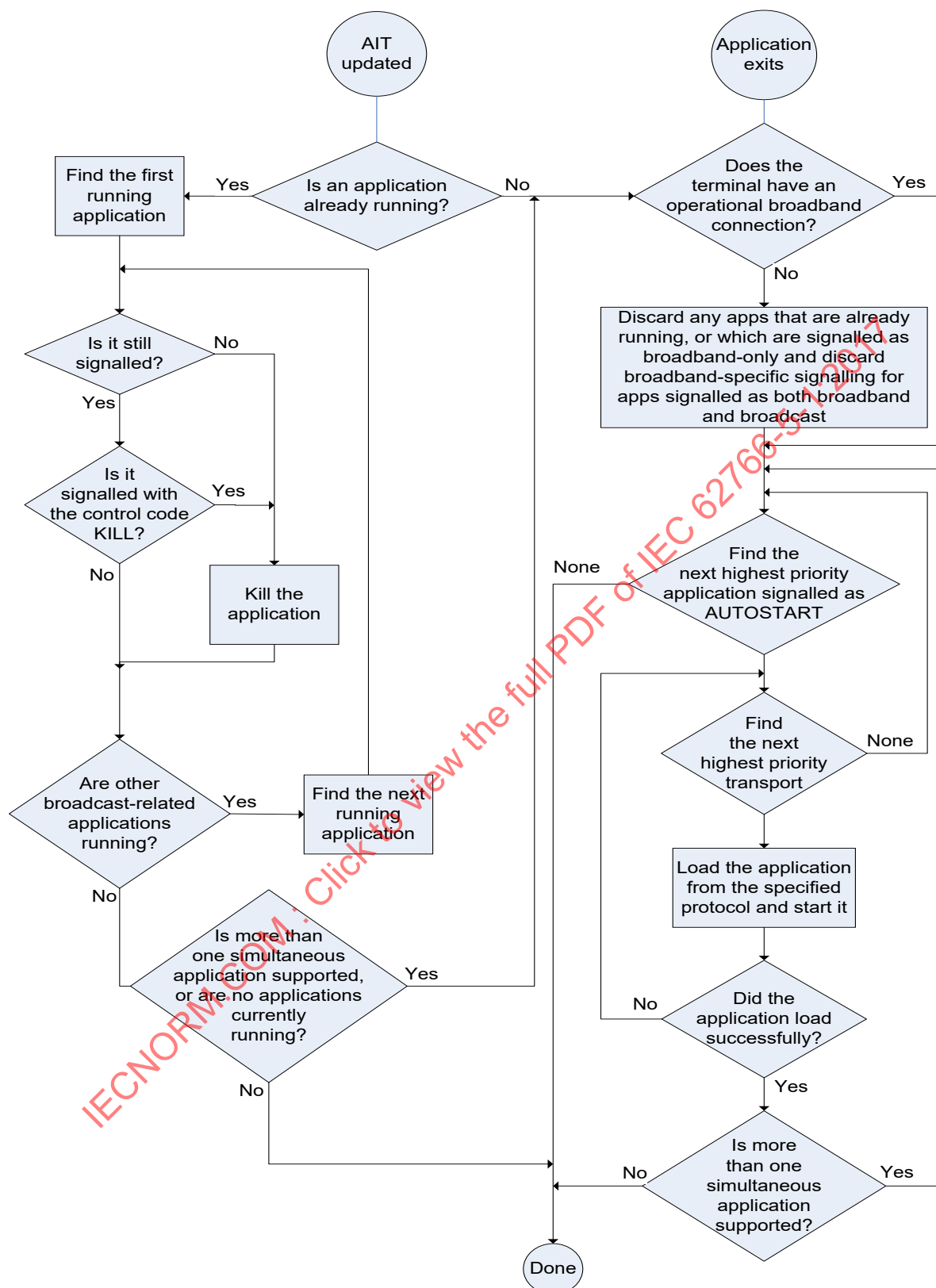
When the application signalling for a scheduled content service is updated, the following apply:

- applications that are added to the service with a control code of AUTOSTART shall be automatically started when their addition is detected by the OITF. They shall be started commencing with the highest priority application working downwards in priority while resources in the OITF permit; applications added to the service with any other control code shall not be automatically started;
- applications that are part of the service whose control code changes to AUTOSTART from some other value shall be automatically started unless already running;
- an application that is removed from the service or whose control code changes to KILL shall be stopped.

If application signalling is removed from a service, all running broadcast-related applications shall be stopped (i.e. the same behaviour as signalling an empty AIT).

Figure 5 shows the behaviour that shall apply when the application signalling for the currently selected service changes, or when a running broadcast-related application exits.

IECNORM.COM : Click to view the full PDF of IEC 62766-5-1:2017



IEC

Figure 5 – Behaviour when the application signalling for the currently selected channel changes or when a running broadcast-related application exits

5.2.4 Service provider related applications

5.2.4.1 Signalling

Service providers may signal service provider related applications as part of their SD&S service provider discovery record (see 4.3.4 of IEC 62766-3:2016, also 4.2.3 and 5.4.3.3 of ETSI TS 102 809 where they are referred to as "unbound applications"). Service provider related applications may either be directly included in the SD&S service provider discovery record or included in an application discovery record that is referenced from the service provider discovery record.

Service providers may label one of the applications in their SD&S service provider discovery record using the application usage values defined in 4.3.4.4.4 of IEC 62766-3 as follows;

- a service discovery application using the ApplicationUsage identifier "urn:oipf:cs:ApplicationUsageCS:2009:servicediscovery". An application labelled in this way should be the highest priority AUTOSTART application signalled;
- an EPG application using the ApplicationUsage identifier "urn:oipf:cs:ApplicationUsageCS:2009:epg";
- a VoD application using the ApplicationUsage identifier "urn:oipf:cs:ApplicationUsageCS:2009:vod";
- a communication service application using the ApplicationUsage identifier "urn:oipf:cs:ApplicationUsageCS:2009:communication";
- an application implementing non-native HNI-IGI using the ApplicationUsage identifier "urn:oipf:cs:ApplicationUsageCS:2009:hni-igi".

5.2.4.2 Starting

Service provider related applications are started under the following circumstances:

- when a service provider is selected, the OITF shall start the AUTOSTART applications signalled by that service provider starting with the one with highest priority and working downwards in priority unless already running, while resources in the OITF permit. This process shall be repeated if an application previously launched by this process is closed for any reason. A consequence of this is that AUTOSTART service provider related applications are always running. Service provider related applications that are not required to be always running shall not be signalled as AUTOSTART themselves, but should be started by AUTOSTART applications;
- by the end-user using a mechanism provided by the OITF;
- by other service provider related applications.

The OITF shall include a mechanism to show the service discovery application and may include mechanisms to show the EPG, VoD and the communication service applications. These mechanisms:

- shall load the application into the browser if not already loaded;
- shall show this application to the end-user;
- shall work at all times when the currently selected service provider has an application labelled in this way.

It is outside the scope of this document whether these mechanisms are buttons on a remote control, items in an on-screen menu or something else. If a button is used, this mechanism shall work regardless of which application has focus and the key event corresponding to the button used shall not be delivered to DAE applications.

5.2.4.3 Stopping

In addition to what is stated in 5.2.2, service provider related applications are stopped when:

- the service provider selection process is re-run and a different service provider is selected;
- the selected service provider updates the list of applications in their SD&S service provider discovery record, an application is removed and the OITF detects this update (see 5.2.2.3 of IEC 62766-3:2016).

5.2.5 Broadcast-independent applications

Applications that are independent of both broadcasters and the currently selected service provider are started and stopped as described in 5.2.2 above. They do not require any signalling. If they are signalled, then this shall be done using the XML encoding of the AIT as defined in 5.4 of ETSI TS 102 809. The XML file shall contain an application discovery record containing exactly one application. The XML file shall be delivered with HTTP or HTTPS using the “application/vnd.dvb.ait+xml” MIME type as defined in 5.4 of ETSI TS 102 809.

5.2.6 Switching between applications

Two cases of switching between applications are relevant in this document:

- switching between visible applications and invisible ones;
NOTE Switching between a visible application and an invisible one is conceptually a little like changing between tabs in a PC browser however without any implication of a particular user interface.
- switching between simultaneously visible applications where this optional feature is supported.

A number of possible mechanisms exist for switching between visible applications and invisible ones. Some examples include the following:

- hard-coded mechanisms in the terminal for switching to a specific application (e.g. to the service discovery application, the content guide, the communication service application);
- an optional terminal specific UI showing available DAE applications which the user can switch to.

5.2.7 Signalling format

5.2.7.1 XML encoding

Table 2 defines how the signalling defined in ETSI TS 102 809 shall be interpreted when used to signal DAE applications.

Table 2 – Application signalling

Descriptor or element	Summary	Status in this document
5.4.4.1 ApplicationList	List of applications	Required
5.4.4.2 Application	Name, identifier, type specific descriptor	Required
5.4.4.3 ApplicationIdentifier	2 numbers	Required
5.4.4.4 ApplicationDescriptor	Numerous application attributes	Required The serviceBound element is only applicable to broadcast related applications and shall be ignored for other applications.
5.4.4.5 VisibilityDescriptor	Attribute – indicate if application can be visible to users and/or other applications	Optional. If this element is not present, OITFs shall use a default value of VISIBLE_ALL.
5.4.4.6 IconDescriptor	Icon for application	The filename in the IconDescriptor shall be an HTTP URL. Use of the icon signalled here by the OITF is optional.

Descriptor or element	Summary	Status in this document
5.4.4.7 AspectRatio	Preferred aspect ratio for icons	Only relevant if the OITF uses the IconDescriptor.
5.4.4.8 MhpVersion	Specification version	As defined in 4.3.4.4.3 of IEC 62766-3:2016.
5.4.4.9 StorageCapabilities	Can the application be stored or cached	Ignored
5.4.4.10 StorageType	Enumeration used in 5.4.4.9 of ETSI TS 102 809	Ignored
5.4.4.11 ApplicationType	Application type	For DAE and PAE applications, the appropriate value from the ApplicationTypeCS scheme from IEC 62766-3 shall be used.
5.4.4.12 DvbApplicationType	Enumeration for 5.4.4.11 of ETSI TS 102 809	Ignored
5.4.4.13 ApplicationControlCode	Enumeration for 5.4.4.4 of ETSI TS 102 809	See below
5.4.4.14 ApplicationSpecificDescriptor	Container	Ignored
5.4.4.15 AbstractIPService	Supports grouping of unbound applications	Only one group shall be signalled
5.4.4.16 ApplicationOfferingType	Used as part of application discovery record	Required
5.4.4.17 ServiceDiscovery	Used as part of application discovery record	Required
5.4.4.18 ApplicationUsageDescriptor	Indicates that an application provides a specific service	Required
5.4.4.19 TransportProtocolDescriptorType	Abstract base type	Required
5.4.4.20 HTTPTransportType	Type for applications accessed by HTTP	Required
5.4.4.21 OCTransportType	Type for applications accessed by DSM-CC object carousel	Ignored
5.4.4.22 ComponentTagType	Encodes a DVB component tag	Ignored
5.4.4.23 SimpleApplicationLocationDescriptorType	Encodes the location of the start page of an application relative to one of the transport types	Required
5.4.4.24 SimpleApplicationBoundaryDescriptorType	Encodes an application boundary	Required
FLUTESessionDescriptor as defined by Clause B.6 of IEC 62766-3:2016	Support for distributing applications through multicast	Shall be supported if OITFs support FLUTE.

Elements and descriptors marked as 'Ignored' shall not be processed for DAE applications. Servers may include these in application signalling.

The application control code shall be interpreted as follows for DAE applications, see Table 3.

Table 3 – DAE application control codes

Value	Description
AUTOSTART	The application is eligible to be started automatically. Subclauses 5.2.3.1 and 5.2.4.1 above define the order in which AUTOSTART applications are started if more than one is signalled.
PRESENT	The OITF shall take no action. The OITF may provide a mechanism to allow the end-user to start applications signalled as PRESENT. However since there is no requirement for such a mechanism, an IPTV service provider who signals applications with this control code shall provide an application able to start them.
KILL	The application shall be terminated (see ApplicationDestroyRequest in 7.2.7).
PREFETCH	The OITF may start fetching files, data or other information needed to start the application but shall NOT start the application. Implementations may consider this control code to be the same as PRESENT.

The other control codes from ETSI TS 102 809 are not defined for DAE applications. Other control codes are not required to be supported but may be supported if required by another specification. The OITF shall discard any AIT entry containing an unsupported control code.

5.2.7.2 MPEG-2 encoding

In a hybrid device where the broadcast channel is based on DVB network technologies and uses DVB-SI as specified in ETSI EN 300 468, the OITF shall support the MPEG-2 encoding of the AIT from ETSI TS 102 809 as defined in Table 4. This encoding may be supported in other devices. Table 5 defines the meaning of entries in the "status" column.

Table 4 – Supported application signalling features

Subclause	Status	Notes
5.2.2 Application types	M	The application type shall be 0x0011.
5.2.3 Application identification	M	Applications that only need the default permissions shall be signalled using application_ids from the range for unsigned applications. Applications that need more permissions than the default shall be signalled using application_ids from the range for signed applications. The range of application_ids for privileged applications shall NOT be used.
5.2.4 Application control codes	M	The following control codes shall be supported: 0x01 AUTOSTART 0x02 PRESENT 0x04 KILL 0x07 DISABLED The application life cycle shall follow the rules defined in ETSI TS 102 809 and in this document.

Subclause	Status	Notes
5.2.5 Platform profiles	M	The encoding of the <code>application_profile</code> is not defined in this document. The version fields shall be set as follows: <code>version.major</code> = 2 <code>version.minor</code> = 3 <code>version.macro</code> = 0
5.2.6 Application visibility	M	
5.2.7 Application priority	M	
5.2.8 Application icons	O	The icon locator information shall be relative to the base part (constructed from the <code>URL_base_bytes</code>) of the URL as signalled in the <code>transport_protocol_descriptor</code> .
5.2.9 Graphics constraints	–	
5.2.10 Application usage	M	Usage type 0x01 shall be supported as described in subclause 5.2.10.2 of ETSI TS 102 809.
5.2.11 Stored applications	–	
5.2.12 Application Description File	–	
5.3.2 Program specific information	M	
5.3.4 Application Information Table	M	See IEC 62766-2-1 for MPEG-2 system related requirements and constraints.
5.3.5.1 Application signalling descriptor	M	
5.3.5.2 Data broadcast id descriptor	O	The value to be used for the <code>data_broadcast_id</code> field of the <code>data_broadcast_id_descriptor</code> for OIPF carousels shall be 0x0150. By supporting this optional feature, terminals can reduce the time needed to mount a carousel.
5.3.5.3 Application descriptor	M	
5.3.5.4 Application recording descriptor	–	
5.3.5.5 Application usage descriptor	M	Usage type 0x01 shall be supported as described in subclause 5.2.10.2 of ETSI TS 102 809.
5.3.5.6 User information descriptors	M	
5.3.5.7 External application authorization descriptor	M	
5.3.5.8 Graphics constraints descriptor	–	
5.3.6 Transport protocol descriptors	M	The following <code>protocol_ids</code> shall be supported: 0x0001 object carousel over broadcast channel (as defined in ETSI TS 102 809) 0x0003 HTTP over broadband connection
5.3.7 Simple application location descriptor	M	
5.3.8 Simple application boundary descriptor	M	Only strict prefixes starting with "dvb://", "http://" or "https://" shall be supported. Only prefixes forming at least a second-level domain shall be supported. Path elements shall be ignored.
5.3.9 Service information	–	
5.3.10 Stored applications	–	

Table 5 – Key to status column

Status	Description
M	MANDATORY The signalling may be restricted to a subset specified in the "Notes" column. In that case all additional signalling is optional.
O	Optional
–	NOT INCLUDED The referenced signalling is not included in this document.

5.2.8 Widgets lifecycle

5.2.8.1 General

As Widgets are packaged as ZIP archives, they only require a single download and installation on an OITF before being executed. Widgets can also be downloaded over non-HTTP distribution channels and even over off-network channels (USB drives, CD/DVD, etc.).

The Widget lifecycle has 3 main steps:

- a) installation: the Widget is installed on the OITF;
- b) execution: the Widget is executed (end eventually stopped);
- c) removal: the Widget is uninstalled from the OITF.

Step a), installation, is only needed before the first execution of the Widget or if its version is obsolete and the user or the OITF wants to update it (see 5.2.8.5).

Step b), execution, may be performed at any time after the Widget has been installed. It can be triggered by an action from the user, or it may be done automatically by the OITF, either through a DAE application or a native application in the OITF. Note that it is not possible to have two running instances of a single Widget simultaneously.

Step c), removal, is performed if the user wants to uninstall the Widget from the OITF. An uninstalled Widget needs to be reinstalled by a user to be executed again.

Detail descriptions of each step above are provided in the following subclauses.

5.2.8.2 Widget installation

In order to be able to execute a Widget, the Widget package first needs to be acquired and installed on the OITF. Steps for acquiring and processing a Widget package and associated processing rules are described in Clause 9 of W3C Widget Packaging and XML Configuration. In this document, the expression "Widget installation succeed" means that the aforementioned procedure is completed successfully.

Although W3C Widget Packaging and XML Configuration does not limit or mandate any specific data transfer protocol or distribution channel through which Widgets are delivered, an OITF shall support the use of HTTP and HTTPS as the transfer protocols. Support for other transfer protocols is optional. Widget installation is done through the `ApplicationManager.installwidget()` API call. After a call to this function, if the installation succeeds, the installed Widget shall be available in the list of installed Widgets that can be retrieved using `ApplicationManager.widgets`. The application installing the Widget is notified about the installation success/failure through the `widgetInstallation` event as specified in 7.2.2.3 and 7.2.2.5.

When installing a Widget, the OITF should notify the user if there is already an installed Widget with the same "id" value (where "id" is defined in 7.6.1 of W3C Widget Packaging and XML Configuration along with the extension defined in 11.2 of this document). In this case, the OITF shall proceed as specified in the description of `installWidget()` method in 7.2.2.4.

5.2.8.3 Widget execution

In order to be executed, a Widget needs to be installed as described in 5.2.8.2. After the installation, a Widget can be started either using the `Application.createApplication()` API call or through the `Application.startWidget()` API call. The behaviour of these two methods is equivalent. `startWidget()` is the preferred method; `createApplication()` is kept for consistency with other DAE applications. A list of installed Widgets can be retrieved using `ApplicationManager.widgets`. Note that only one running instance per Widget at a time is allowed. A Widget can be stopped using `Application.stopWidget()` or `Application.destroyApplication()`. `stopWidget()` is the preferred method; `destroyApplication()` is kept for consistency with other DAE applications.

If the installed Widget has been run on the OITF before, any "storage areas" associated with the Widget, as defined in W3C Widget Interface, shall be restored. Saved data is accessible through the preferences attribute of the Widget object as defined in 11.4.

See related subclauses in Clause 7 for more details about the above-mentioned API calls.

5.2.8.4 Uninstalling a widget

An installed Widget can be uninstalled from an OITF through the `ApplicationManager.uninstallWidget()` API call. Calling this method on a running Widget will cause the Widget to be stopped before the Widget is uninstalled. The application uninstalling the Widget is notified about the uninstallation success/failure through the "WidgetUninstallation" event as specified in 7.2.2.3 and 7.2.2.5. Any storage areas associated with the uninstalled Widget shall be deleted.

5.2.8.5 Widget updates

An installed widget can be updated by installing a new version of it.

5.3 Event notifications

5.3.1 General

Subclause 5.3 describes 4 different notification frameworks (in-session notification based on the home network domain, in-session notification based on the Internet domain, 3rd party notification based on home network domain, and 3rd party notification based on the Internet domain) defined by CEA-2014-A. Moreover, it defines a new notification framework for IMS-based notifications such as CallerID, Incoming Call Message, and Chat Invite not only when a DAE application is active, but also inactive.

The event notification mechanism allows OITFs to receive important UI updates or information from IPTV service provider or home network devices such as IG, AG or DLNA RUI compatible devices. CEA 2014 mandates 4 unique notification models that are dependent on whether the server exists on the internet domain or home network domain. Each of these domain models have two unique scenarios depending on whether or not a DAE application is running. If a DAE application is active, the in-session notifications are used to support dynamic UI interaction between the server and the DAE application without the need to reload the XHTML page. Otherwise, 3rd party event notifications should be used to receive and display a notification message outside of the current user session with a DAE application on the OITF, for example an event coming from another server, e.g. to receive emergency alerts, or events regarding news, weather, stock or other information. Generally, 3rd party event notification creates a new DAE application to display notification information.

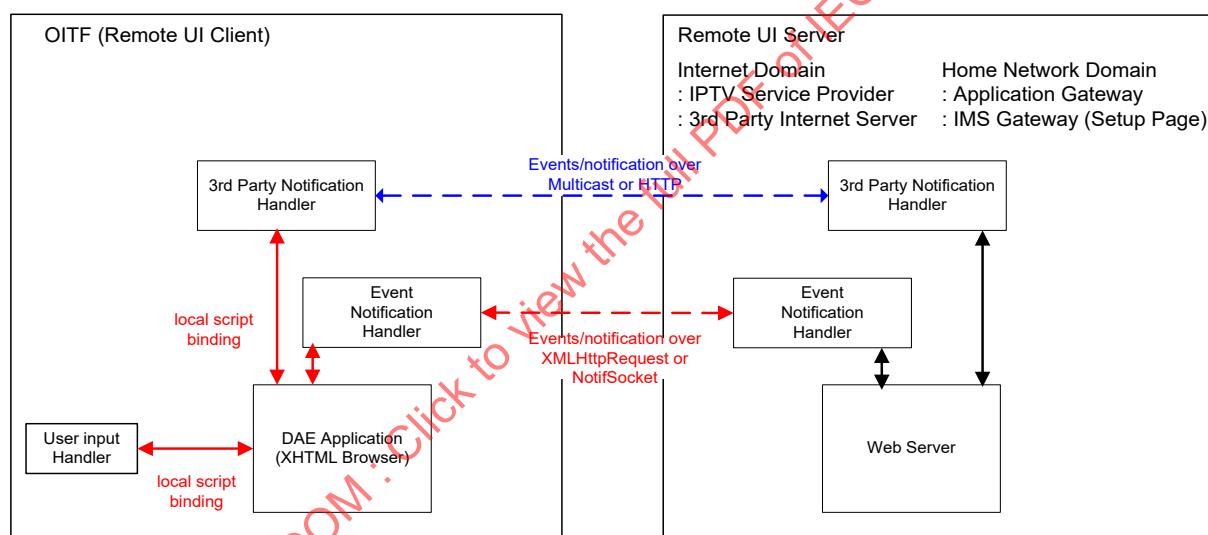
IMS event notifications for Caller ID, Messaging and Chatting have different behaviour from general event notification defined by CEA-2014-A because IMS communication service should be accessed by authorized users and devices within the approval of IPTV service provider. Considering the issue of user's privacy, the DAE specification not only adopts the general Event Notification Frameworks from CEA-2014-A as defined in 5.3.2, but also defines a new IMS Event Notification Framework in 5.3.3.

5.3.2 Event notification framework based on CEA 2014

5.3.2.1 General

An OITF shall be capable of displaying various event notifications from both the Internet domain and the home network domain. Event notification can be conveyed through the active UI interaction's channel or out of session. As described in the diagram below, in-session notification is associated with a running DAE application, whereas a 3rd party event notification is delivered through an independent communication channel. If an OITF receives a 3rd party event after subscribing to a certain internet URL or the OITF receives a multicasted event notification message, the OITF needs to perform 3rd party event notification and display its information inside a new DAE application.

Figure 6 describes a general overview of Event Notification architecture.



IEC

Figure 6 – General event notification architecture on OITF and remote UI server

In-session notifications are performed to update partial or whole DAE application UI through the `NotifSocket` object and/or the `XMLHttpRequest` object as defined by CEA-2014-A. The `NotifSocket` object creates a persistent TCP connection between a DAE application and remote UI server in order to support burst event notifications. In addition, a DAE application can create an `XMLHttpRequest` object to make asynchronous HTTP requests to a web server on the internet domain. This establishes an independent HTTP connection channel to support XML updates between the DAE application and the remote UI server.

On the other hand, if the OITF receives an incoming notification outside of an active interaction (i.e. session) with the server, a 3rd party Event Notification shall be executed to invoke a DAE application to fetch and render the UI content using the URL contained within the notification message. This allows servers to "broadcast" important messages, such as emergency alert messages, to an OITF at any time, even when the DAE application would currently not be running. This should be done through a push-method with multicast message for the home network domain and a pull-method for the internet case.

Subclauses 5.3.2.2 and 5.3.2.3 describe the requirements for the event mechanisms in more detail.

5.3.2.2 In-session event notification

In-session notification can be defined as "Dynamic UI Update." With this mechanism, a server should be able to send a notification message during a UI interaction to update the UI dynamically without the need to reload the XHTML-page. The OITF shall support the two following scripting objects for in-session event notification:

- XMLHttpRequest Scripting Object (as defined in the XMLHttpRequest specification as referenced in IEC 62766-5-2):
 - The XMLHttpRequest is an embedded object on the browser and enables scripts to make HTTP request to a web server without the need to reload the page. It can be used by JavaScript to transfer and manipulate XML data to and from a web server using HTTP, establishing an independent connection channel between a web server and DAE applications. Whenever a DAE application needs to update the UI, it sends a request to the UI server, IPTV service provider or 3rd party Internet Server, to monitor the change of status or event. In case an event, the UI server sends an HTTP response to the XMLHttpRequest.
- NotifSocket Scripting Object (as defined in 5.5.1 CEA-2014-A):
 - Even though support for the XMLHttpRequest object has become more widespread on browsers and Internet Portal servers, it has a difficulty in supporting dynamic UI update on home domain's devices because it is required to be invoked by the request of XMLHttpRequest on DAE application side. NotifSocket creates a persistent TCP connection between DAE application and UI server in order to support burst event notifications. Whenever the UI server needs to notify the DAE application running on the OITF of a UI update, it sends any types of update message, such as encoded binary or string, through the NotifSocket connection. The NotifSocket object allows an UI server to push any event information through the independent TCP/IP channel at any time.

5.3.2.3 Out of session event notification

Out of session event notifications are defined as "3rd party Notifications" in CEA-2014. Since these notifications are not part of an active remote UI interaction with a remote UI Server, the OITF shall launch a new DAE application to render the UI content using the URL contained within the notification message.

The OITF shall support multicast notifications for 3rd party event notifications for the home network domain and the internet domain respectively as defined below. Support for polling-based notifications as defined below is optional and support can be indicated through the OITF's capability description by using element <pollingNotifications> as defined in 9.3.15 or the +POLLNOTIF name fragment as defined in 9.2.

- Multicast Notifications (as defined in 5.6.1 of CEA-2014-A)
 - The OITF shall support receiving of Multicast Notifications over multicast UDP, with a UPnP event message format defined by CEA 2014 if the incoming message comes from home network domain. After interpreting the message, the OITF should create a new notification window with specified <ruieventURL>. In order to ensure a reliable transmission of a multicast notification message, a remote UI Server shall transmit the same notification message, with the same HTTP SEQ header value 2 or 3 times, where the time between transmissions should be a random time between 0 s and 10 s.
- Polling-based Notification (as defined in 5.6.2 of CEA-2014-A) and Annex M in this document.
 - The OITF shall support polling-based 3rd party notifications from an IPTV Service Provider or a 3rd party Internet Server. To this end, the OITF subscribes to certain URIs to display web contents such as news, weather, stock or other information from Internet side on executing the **subscribeToNotifications()** method. An OITF should

poll for notifications even when the CE-HTML browser is not active. If a new notification is received, this may be notified to the user in a vendor defined way, including direct rendering on the display and using a non-intrusive prompt.

NOTE Annex M defines a `subscribeToNotificationsAsync()` method to provide a way of subscribing to polling-based notifications that is non-blocking.

5.3.3 IMS event notification framework

5.3.3.1 General

Subclause 5.3.3 covers the DAE interactions needed to drive the message exchanges on the HNI-IGI interface in the case where the Service Provider offers an IMS application.

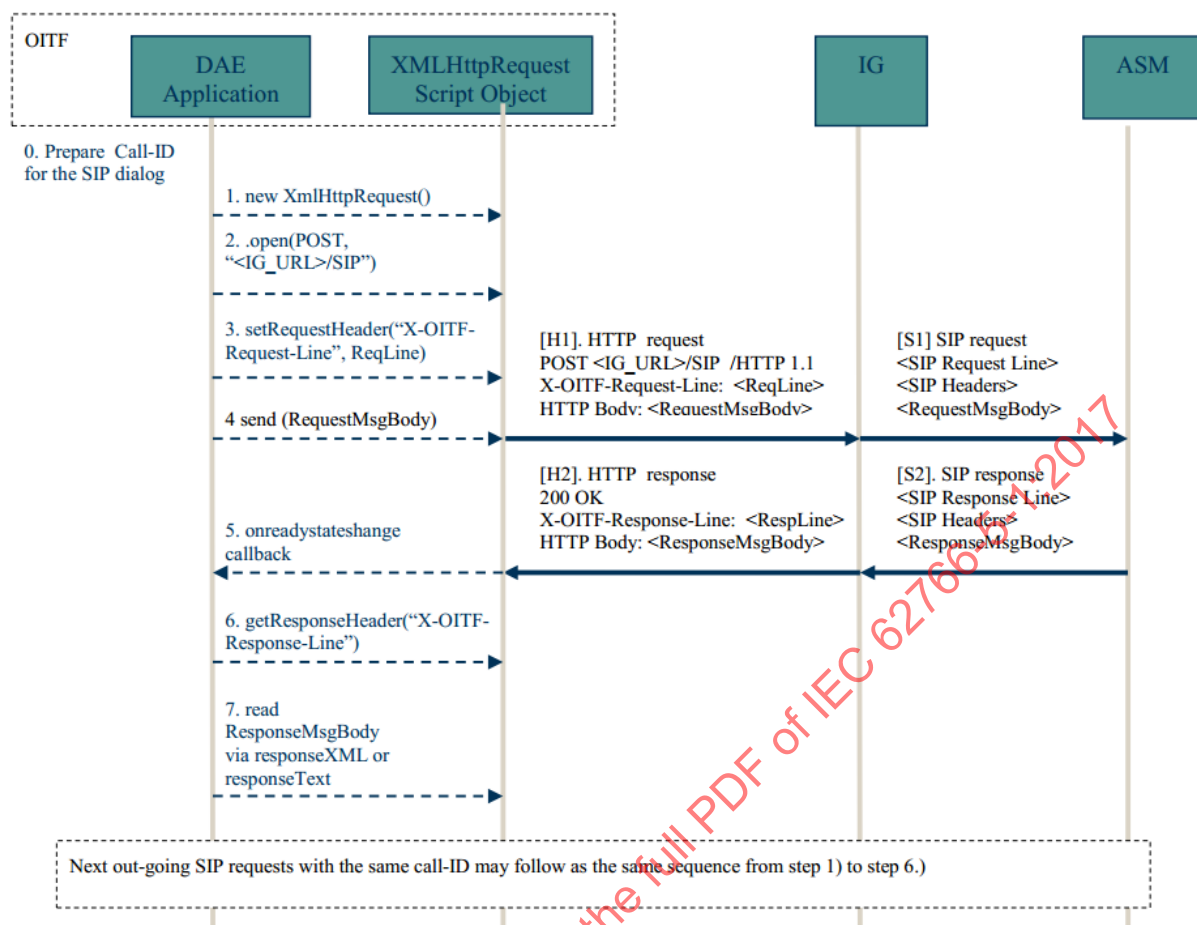
The HNI-IGI framework defines how an OITF interacts with an IMS Gateway (IG) via the HNI-IGI interface (IEC 62766-4-1:2017, 6.2).

Every message on the HNI-IGI interface shall be carried in a HTTP transaction where the OITF sends the HTTP request and the IG responds to the request. The HNI-IGI in-session framework, in the case of a DAE application, uses the XMLHttpRequest Script Object, as defined in the XMLHttpRequest specification as referenced in IEC 62766-5-2.

There are two message directions on the HNI-IGI interface, corresponding to outgoing and incoming messages from and to the OITF.

5.3.3.2 HNI-IGI transactions for in-session out-going request messages

This message direction applies to outgoing messages from the OITF on the HNI-IGI interface. The OITF sends a request and the IG responds to the request. Figure 7 illustrates the sequences for in-session transactions for outgoing requests from DAE application to the IG.



IEC

Figure 7 – HNI-IGI transaction for outgoing SIP requests from a DAE application

The steps in Figure 7 are described below:

Step 0: prepare the Call-ID for a SIP request. The Call-ID shall be generated by the DAE application for an outgoing SIP request. This Call-ID shall be locally unique across all OITFs in a residential network.

NOTE How uniqueness is achieved is currently not defined.

Step 1: the DAE application shall create a new XMLHttpRequest object using the constructor "new XMLHttpRequest()".

Step 2: the DAE application shall invoke the open() method to specify the HTTP method and Request-URI for the request. In this case, the HTTP POST method with the Request-URI of <IG_URL>/SIP shall be used as specified in IEC 62766-4-1.

Step 3: the DAE application shall invoke the setRequestHeader() method to specify the required HTTP headers as specified in IEC 62766-4-1. This method shall be invoked for each required HTTP header. For example, the X-OITF-Request-Line HTTP header specifies the SIP request line for the SIP request. The Call-ID is specified in the X-OITF-Call-ID header.

Step 4: the DAE application shall invoke the send() method to send the HTTP request. The SIP Message Request body is specified in a parameter of this method.

Step 5: when the HTTP response is received, the onreadystatechange callback function shall be invoked on the DAE application.

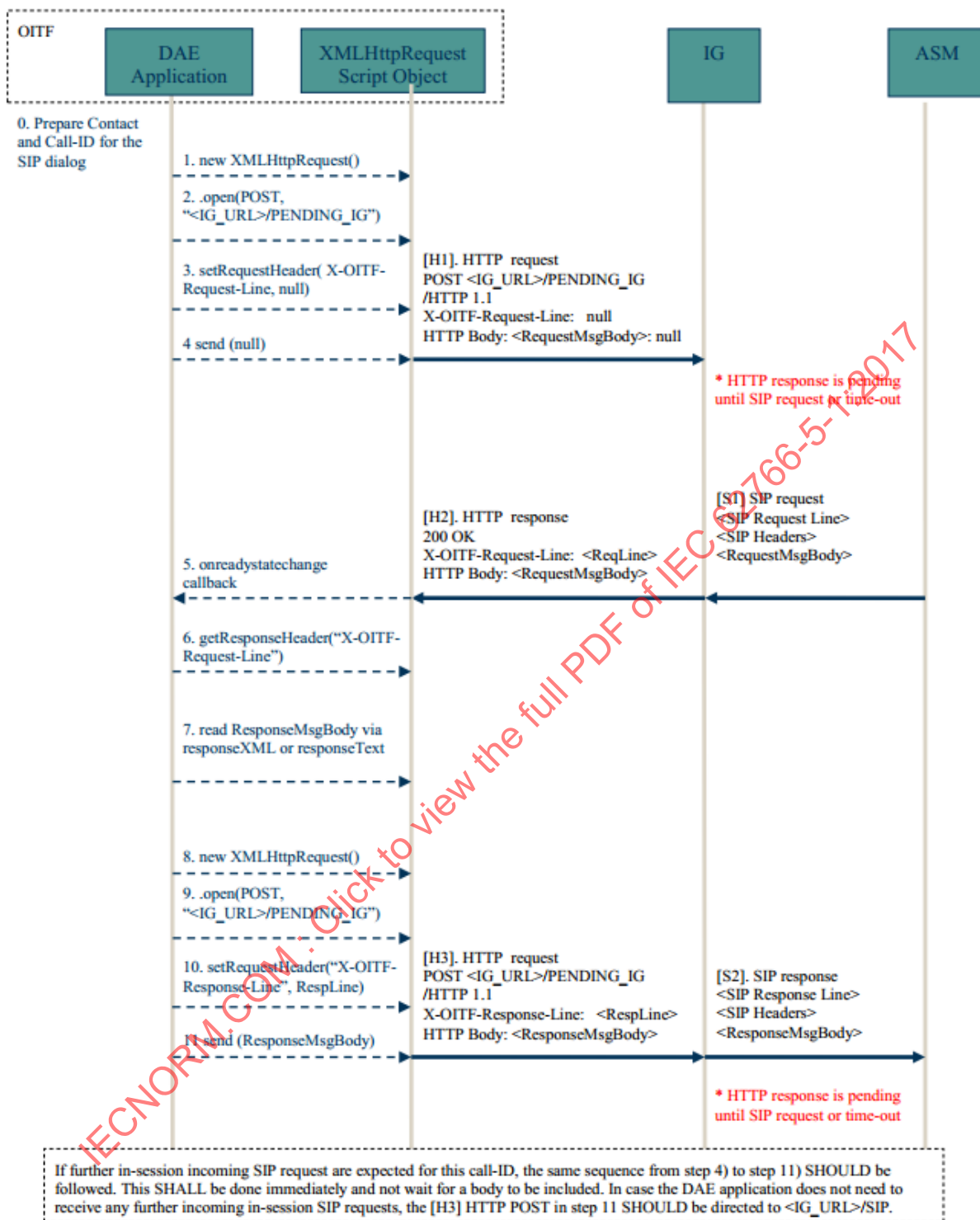
Step 6: the DAE application shall invoke the `getRequestHeader()` method to retrieve each HTTP header. The SIP Response Line is specified in the X-OITF-Response-Line header.

Step 7: if the `readyState` property of the `XMLHttpRequest` object has the value 4, the HTTP response body shall be retrieved via the `responseXML` or `responseText` properties of the `XMLHttpRequest` object. The SIP response body is specified in the HTTP response body.

5.3.3.3 HNI-IGI transaction for in-session incoming request messages

This message direction applies to incoming messages to the OITF on the HNI-IGI interface, which are related to an existing IMS session. An example of this is a SIP NOTIFY message received from the network in response to a previous SIP SUBSCRIBE sent from the IG. The OITF sends a HTTP request and the IG responds to the request when it receives an incoming message from the network related to an existing session. Figure 8 illustrates the sequences for in-session transactions for incoming requests from the IG to the DAE application.

IECNORM.COM : Click to view the full PDF of IEC 62766-5-1:2017



IEC

Figure 8 – HNI-IGI transaction for in-session incoming SIP request

The steps in Figure 8 are described hereafter:

Step 0: prepare the Call-ID for this SIP session for which a message is expected. The Call ID shall be the same as the one created initially for this session.

Step 1: the DAE application shall create a new XMLHttpRequest object using the constructor "new XMLHttpRequest()".

Step 2: the DAE application shall invoke the `open()` method to specify the HTTP method and the Request-URI for the request. In this case, the POST method with a Request-URI of `<IG URL>/PENDING_IG` shall be used as specified in IEC 62766-4-1.

Step 3: the DAE application shall invoke the `setRequestHeader()` method to specify the required HTTP headers, as specified in IEC 62766-4-1. This method is invoked for each HTTP header that is required. In this case, the X-OITF-Request-Line, which specifies the SIP request line for the SIP request, is set to the value `null`. The SIP Call-ID is specified in the X-OITF-Call-ID header.

Step 4: the DAE application shall invoke the `send()` method to send the HTTP request. For the HTTP request that sets up the initial long poll, no X-OITF headers are allowed for the HTTP request to the `PENDING_IG` Request-URI.

Step 5: when the HTTP response is received, the specified `onreadystatechange()` callback function is invoked.

Step 6: the DAE application shall invoke the `getResponseHeader()` method to retrieve each HTTP header. The SIP Request Line is specified in the X-OITF-Request-Line HTTP header.

Step 7: if the `readyState` property of the `XMLHttpRequest` object has the value 4, the HTTP response body shall be retrieved via the `responseXML` or `responseText` properties of the `XMLHttpRequest` object. The SIP response body is specified in the HTTP response body.

Step 8: the DAE application shall create a new `XMLHttpRequest` object using the constructor `"new XMLHttpRequest()"`.

Step 9: the DAE application shall invoke the `open()` method to specify the HTTP method and the Request-URI for the request. In this case, the POST method with a Request-URI of `<IG URL>/PENDING_IG` shall be used as specified in IEC 62766-4-1.

Step 10: the DAE application shall invoke the `setRequestHeader()` method to populate each HTTP header as specified in IEC 62766-4-1. This method shall be invoked for each required HTTP header. For example, the X-OITF-Response-Line specifies the SIP response line for the SIP response. The Call-ID is specified in the X-OITF-Call-ID header.

Step 11: the DAE application shall invoke the `send()` method to send the HTTP request. If there is a SIP response body, it is included as a parameter to the `send()` method. The SIP response body message is carried in the HTTP body for the HTTP request to the `PENDING_IG` Request-URI.

In the case where the OITF does not need to receive any further incoming in-session SIP requests, the HTTP POST in step 11 shall be directed to the `<IG_URL>/SIP` Request-URI.

5.3.3.4 HNI-IGI transaction for out of session incoming request messages

This message direction applies to incoming messages on the HNI-IGI interface that are not related to an existing session. An example of this is a SIP MESSAGE message received from the network, coming for example from an IPTV application or from another user. Figure 9 illustrates the sequences of out-of-session transactions for in-coming requests from the IG to OITF.

Figure 9 describes what happens when the OITF is first turned on.

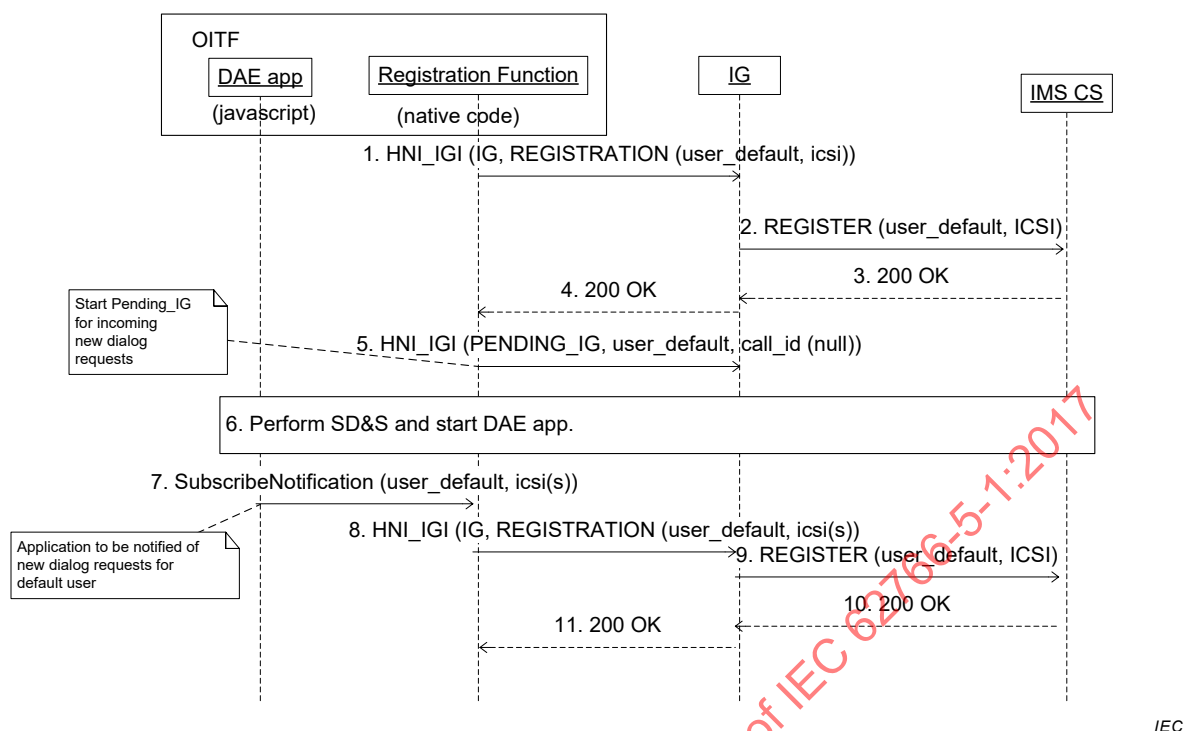


Figure 9 – What happens when the OITF is first turned on

The steps in Figure 9 are described below:

Step 1: when the OITF is turned on the OITF shall send a HNI_IGI IG registration message to register the default user.

Step 2: the IG Registers the default user in the IMS network.

Step 3: the IMS network returns 200 OK.

Step 4: a 200 OK message shall be returned on the HNI_IGI.

Step 5: if there are native IMS applications that may receive unsolicited messages the OITF shall send a PENDING_IG message to the IG, for the default user and with the call_id set to null. The steps to send PENDING_IG are the same as steps 8-11 from 5.3.3.3.

Step 6: the OITF performs service selection and discovery and loads the initial DAE page.

Step 7: DAE IMS applications that desires to receive unsolicited notifications shall issue a subscribeNotification() method (as defined in 7.8.2.3).

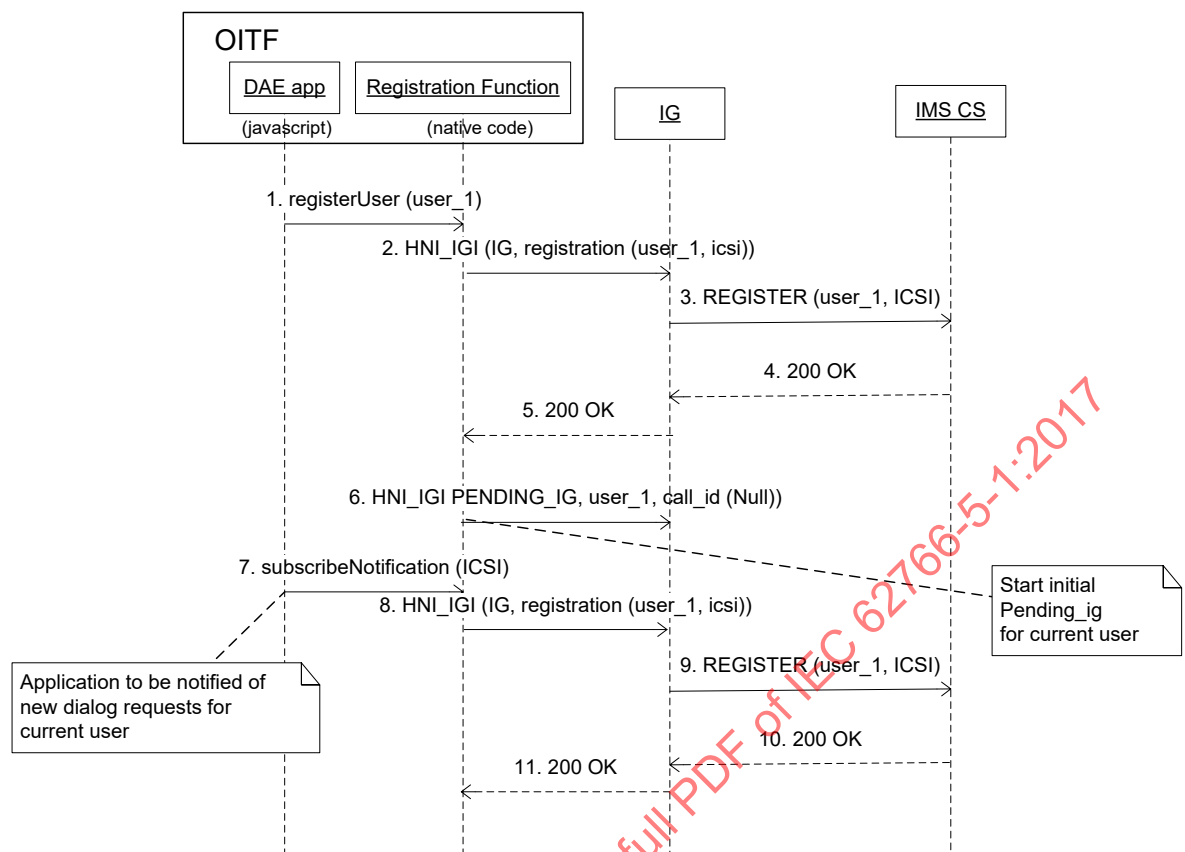
Step 8: when applicable, the OITF shall send a HNI_IGI IG registration message to re-register the default user, including new applications.

Step 9: the IG re-registers the default user in the IMS network.

Step 10: the IMS network returns 200 OK.

Step 11: a 200 OK message shall be returned on the HNI_IGI.

Figure 10 describes what happens when a specific user logs in using the DAE interface.



IEC

Figure 10 – User logs in using the DAE interface

The steps in Figure 10 are described below:

Step 1: when the user desires to login the DAE shall call the `registerUser()` method to register the user.

Step 2: the OITF shall send a `HNI_IGI` IG registration message to register the user.

Step 3: the IG Registers the user in the IMS network.

Step 4: the IMS network returns 200 OK.

Step 5: a 200 OK message shall be returned on the `HNI_IGI`.

Step 6: if there are native IMS applications that may receive unsolicited messages the OITF shall send a `PENDING_IG` message to the IG, for the default user and with the `call_id` set to null. The steps to send `PENDING_IG` are the same as steps 8-11 from 5.3.3.3.

Step 7: DAE IMS applications for the user that desires to receive unsolicited notifications shall issue a `subscribeNotification(icsi)` method (as defined in 7.8.2.3).

Step 8: when applicable, the OITF shall send a `HNI_IGI` IG registration message to re-register the user, including new applications.

Step 9: the IG re-registers the default user in the IMS network.

Step 10: the IMS network returns 200 OK.

Step 11: a 200 OK message shall be returned on the HNI_IGI.

Figure 11 describes what happens when an unsolicited message arrives from the network. The precondition is that a DAE application is already running and subscribed to the IMS notifications (refer to previous sequence when user logs in).

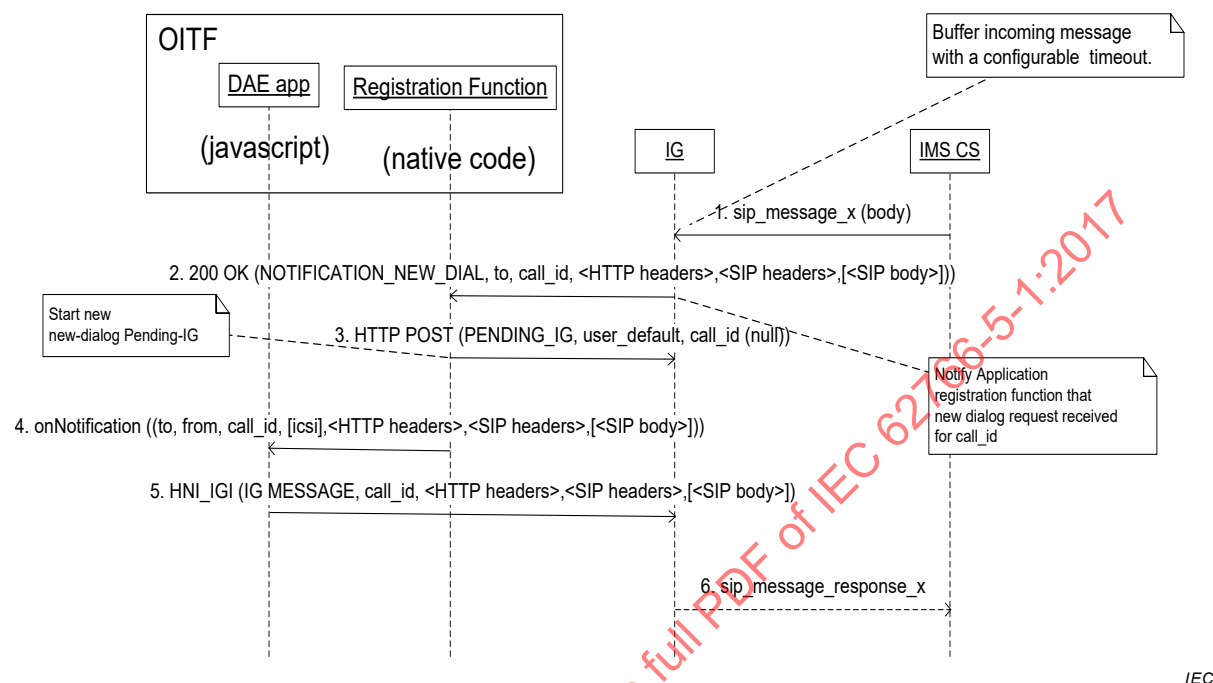


Figure 11 – Unsolicited message from the network

The steps in Figure 11 are described below:

Step 1: a SIP message arrives from the network.

Step 2: the IG responds to the PENDING_IG request.

Step 3: the OITF shall immediately issue a new PENDING_IG request after receiving a response on a PENDING_IG request. The steps to send PENDING_IG are the same as steps 8-11 from 5.3.3.3.

Step 4: the OITF shall call the callback function onNotification for the corresponding application. This includes the IMS message.

Step 5: the OITF may respond to the network with a new outgoing message. The steps to send PENDING_IG are the same as steps 8-11 from 5.3.3.3.

Step 6: if the OITF sends a message the IG shall forward it to the network.

6 Formats

6.1 Web standards TV profile

6.1.1 General

An OITF shall support the Web Standards TV Profile defined in IEC 62766-5-2. The MIME type used for DAE documents shall be one of the MIME types defined in the HTML5

specification as referenced by IEC 62766-5-2 for HTML or XHTML documents, or the value 'application/xhtml+xml; charset="UTF8"'.

6.1.2 Additional restrictions and requirements

The following properties and methods shall be supported on the window object as defined in subclause "The window Object" of the HTML5 specification as referenced in IEC 62766-5-2 with additional requirements relating to integration with APIs and mechanisms defined in this document:

- close(): calling this method on the Window object of a DAE application shall be equivalent to calling method destroyApplication() of the DAE application (as defined in 7.2.3.3);
- blur(): calling this method on the Window object of a DAE application shall not deactivate the application as defined in 4.4.9;
- window scripting object support for the additional methods defined in Annex M is indicated by the <pollingNotifications> element in the device capabilities as defined in 9.3.15;
- an OITF shall offer a means to set focus to the following elements in a HTML document by using key-based input: <a>, <area>, all form elements, <iframe>, and <object> elements of type "video". These elements shall have the tabIndex focus flag set (see 7.4.1 of the HTML5 specification as referenced by IEC 62766-5-2);
- an OITF shall allow the user to manually trigger elements that have an activation behaviour, for instance using keyboard input.

NOTE Subclause 3.2.5.1.7 ("Interactive content") of the HTML5 specification as referenced by IEC 62766-5-2 includes similar requirements, however using "should" not "shall".

6.2 Still image formats

The following still image formats shall be supported:

- GIF as defined in Graphics Interchange Format;
- PNG as defined in ISO/IEC 15948:2004;
- JPEG as defined in JPEG File Interchange Format, except that support for lossless and hierarchical modes and arithmetic coding of DCT coefficients is optional. The thumbnail feature of JPEG File Interchange Format is optional. OITFs not supporting thumbnails shall skip them if present and continue decoding the rest of the image.

6.3 Media formats

6.3.1 General

Subclause 6.3 describes the main requirements for the format and usage of codecs in media referred to by DAE applications. This subclause also describes memory audio.

6.3.2 Media format of A/V media except for audio from memory

This subclause describes the format and usage of the A/V media codec except for audio from memory.

- Format and usage of video codecs shall adhere to Clause 6 of IEC 62766-2-1:2016.
- The format and usage of subtitle streams shall adhere to Clause 7 of IEC 62766-2-1:2016.
- The format and usage of teletext information shall adhere to Clause 8 of IEC 62766-2-1:2017.
- The format and usage of audio codecs shall adhere to Clause 9 of IEC 62766-2-1:2016, except for 9.1.1.2, 9.1.5 and 9.2.1 which are covered in 6.3.3 of this document.

6.3.3 Media format of A/V media for audio from memory

This subclause describes the format and usage of the A/V media codec for audio from memory. Usage of corresponding A/V media object is described in 7.14 .

For the audio from memory format, HE-AAC shall be supported by the OITF and WAVE may be supported by the OITF.

- The format and usage of HE-AAC audio from memory shall adhere to 9.2.1.3 and 9.3.1 of IEC 62766-2-1:2016.
- The format and usage of WAVE audio from memory shall adhere to 9.2.6 and 9.3.1 of IEC 62766-2-1:2016.

6.3.4 Media transport

The format and usage of media transports referred to by DAE applications shall adhere to Clause 5 of IEC 62766-2-1:2016.

6.4 SVG

Integration between SVG and HTML is defined in the HTML5 specification as referenced in IEC 62766-5-2.

7 APIs

7.1 Object factory API

7.1.1 General

Subclause 7.1 defines the methods to check and create an instance of the DAE defined embedded objects within JavaScript.

The OITF shall support an object of type "oipfObjectFactory" as a property "oipfObjectFactory" of the script's global object (as defined in the HTML5 specification as referenced by IEC 62766-5-2) with the API as defined in 7.1. The object factory shall ensure that the referenced objects are correctly set up. This is an alternative to instantiating embedded objects (or plug-ins) outside of JavaScript.

7.1.2 Methods

7.1.2.1 General

Boolean isObjectSupported(String mimeType)																						
Description	This method shall return true if and only if an object of the specified type is supported by the OITF, otherwise it shall return false .																					
Arguments	mimeType	If the value of the argument is one of the MIME types defined in Tables 1 to 4 of IEC 62766-2-1:2016 or one of the DAE defined mime types listed below, then an accurate indication of the OITF's support for that MIME type shall be returned. For other values, it is recommended that an OITF returns a value which accurately reflects its support for the specified MIME type. <table><tr><th>DAE MIME Type</th></tr><tr><td>application/notifsocket</td></tr><tr><td>application/oipfApplicationManager</td></tr><tr><td>application/oipfCapabilities</td></tr><tr><td>application/oipfCodManager</td></tr><tr><td>application/oipfCommunicationServices</td></tr><tr><td>application/oipfConfiguration</td></tr><tr><td>application/oipfDownloadManager</td></tr><tr><td>application/oipfDownloadTrigger</td></tr><tr><td>application/oipfFormAgent</td></tr><tr><td>application/oipfGatewayInfo</td></tr><tr><td>application/oipfCommunicationServices</td></tr><tr><td>application/oipfMDTF</td></tr><tr><td>application/oipfParentalControlManager</td></tr><tr><td>application/oipfRecordingScheduler</td></tr><tr><td>application/oipfRemoteControlFunction</td></tr><tr><td>application/oipfRemoteManagement</td></tr><tr><td>application/oipfSearchManager</td></tr><tr><td>application/oipfStatusView</td></tr><tr><td>video/broadcast</td></tr></table>	DAE MIME Type	application/notifsocket	application/oipfApplicationManager	application/oipfCapabilities	application/oipfCodManager	application/oipfCommunicationServices	application/oipfConfiguration	application/oipfDownloadManager	application/oipfDownloadTrigger	application/oipfFormAgent	application/oipfGatewayInfo	application/oipfCommunicationServices	application/oipfMDTF	application/oipfParentalControlManager	application/oipfRecordingScheduler	application/oipfRemoteControlFunction	application/oipfRemoteManagement	application/oipfSearchManager	application/oipfStatusView	video/broadcast
DAE MIME Type																						
application/notifsocket																						
application/oipfApplicationManager																						
application/oipfCapabilities																						
application/oipfCodManager																						
application/oipfCommunicationServices																						
application/oipfConfiguration																						
application/oipfDownloadManager																						
application/oipfDownloadTrigger																						
application/oipfFormAgent																						
application/oipfGatewayInfo																						
application/oipfCommunicationServices																						
application/oipfMDTF																						
application/oipfParentalControlManager																						
application/oipfRecordingScheduler																						
application/oipfRemoteControlFunction																						
application/oipfRemoteManagement																						
application/oipfSearchManager																						
application/oipfStatusView																						
video/broadcast																						

7.1.2.2 Visual objects

The methods in this subclause all return `HTMLObjectElement` objects which can be inserted into the DOM tree. All objects in Clause 7 which have a visual representation on the screen can be created using methods in this subclause. Only for objects defined in Clause 7, that are supported by the device (i.e. as indicated through the client capability description), a corresponding method name to instantiate the object through the `OipfObjectFactory` class can be assumed to be present on the `oipfObjectFactory` object. For any other object, a corresponding method name cannot be assumed to be present.

HTMLObjectElement createVideoBroadcastObject (StringCollection requiredCapabilities)		
HTMLObjectElement createVideoMpegObject (requiredCapabilities) StringCollection		
HTMLObjectElement createStatusViewObject ()		
Description	If the object type is supported, each of these methods shall return an instance of the corresponding embedded object. Since objects do not claim scarce resources when they are instantiated, instantiation shall never fail if the object type is supported. If the method name to create the object is not supported, the OITF shall throw an error with the error.name set to the value "TypeError". If the object type is supported, the method shall return an HTMLObjectElement equivalent to the specified object. The value of the type attribute of the HTMLObjectElement shall match the mimetype of the instantiated object, for example "video/broadcast" in case of method oipfObjectFactory.createVideoBroadcastObject().	
Arguments	requiredCapabilities	An optional argument indicating the formats to be supported by the resulting player. Each item in the argument shall be one of the formats specified in IEC 62766-2-1. Scarce resources will be claimed by the object at the time of instantiation. The allocationMethod property shall be set STATIC_ALLOCATION. If the OITF is unable to create the player object with the requested capabilities, the method shall return null. If this argument is omitted, objects do not claim scarce resources so instantiation shall never fail if the object type is supported. The allocationMethod property shall be set to DYNAMIC_ALLOCATION.

7.1.2.3 Non-visual objects

The methods in this subclause all return JavaScript objects which implement the interfaces of their corresponding objects. They cannot be inserted in the DOM tree. All objects in Clause 7 that do not have a visual representation on the screen can be created using methods in this subclause. Only for objects defined in Clause 7, that are supported by the device (i.e. as indicated through the client capability description), a corresponding method name to instantiate the object through the `OipfObjectFactory` class can be assumed to be present on the `oipfObjectFactory` object. For any other object, a corresponding method name cannot be assumed to be present.

<pre> Object createApplicationManagerObject () Object createCapabilitiesObject () ChannelConfig createChannelConfig () Object createCodManagerObject () Object createConfigurationObject () Object createDownloadManagerObject () Object createDownloadTriggerObject () Object createDrmAgentObject () Object createGatewayInfoObject () Object createIMSObject () Object createMDTFObject () Object createNotifSocketObject () Object createParentalControlManagerObject () Object createRecordingSchedulerObject () Object createRemoteControlFunctionObject () Object createRemoteManagementObject () Object createSearchManagerObject () </pre>	
Description	If the object type is supported, each of these methods shall return an instance of the corresponding embedded object. This may be a new instance or existing instance. For example, the object will likely be a global singleton object and calls to this method may return the same instance. Since objects do not claim scarce resources when they are instantiated, instantiation shall never fail if the object type is supported. If the method name to create the object is not supported, the OITF shall throw an error with the <code>name</code> property set to the value <code>"TypeError"</code>. If the object is supported, the method shall return a JavaScript <code>Object</code> which implements the interface for the specified object.

7.1.3 Examples

This subclause provides examples of the usage of the methods.

The first example shows how to query whether an instance of the A/V Control object for a specified MIME type can be created without the application having to attempt to instantiate the object.

```

var videoPlayer;
if (window.oipfObjectFactory.isObjectSupported("video/mpeg")) {
    videoPlayer = window.oipfObjectFactory.createVideoMpegObject();
    // append object to document
    document.getElementById('playerDiv').appendChild(videoPlayer);
    videoPlayer.data = "rtsp://server/barker_channel";
}

```

If the OITF does not support the created object, the OITF shall throw an error with the error.name set to the value "TypeError". The example below shows how this can be used by applications:

```

try {
    configuration = window.oipfObjectFactory.createConfigurationObject();
}
catch (error) {
    alert("application/oipfConfiguration object could not be created - error
name: " +
        error.name + " - error message: " + error.message);
}

```

7.2 Application management APIs

7.2.1 General

An OITF providing DAE application capability shall implement the behaviour of the classes defined in 7.2.

7.2.2 The application/oipfApplicationManager embedded object

7.2.2.1 General

An OITF shall support a non-visual embedded object of type "application/oipfApplicationManager", with the following JavaScript API, to enable applications to access the privileged functionality related to application lifecycle and management that is provided by the application model defined in 7.2.2.

If one of the methods on the application/oipfApplicationManager is called by a webpage that is not a privileged DAE application, the OITF shall throw an error as defined in 10.1.2.

7.2.2.2 Constants

The following constants are defined as properties of the application/oipfApplicationManager embedded object:

Name	Value	Use
WIDGET_INSTALLATION_STARTED	0	The Widget installation has started. This state shall be used to indicate that the download of the Widget package is completed (possibly because the Widget was already stored locally) and the OITF is ready to start the Widget installation process. This state shall NOT be signalled if the package download fails.
WIDGET_INSTALLATION_COMPLETED	1	The Widget installation has completed successfully
WIDGET_INSTALLATION_FAILED	2	The Widget installation has failed either because the Widget package download failed or because, after the download, the Widget installation process failed.
WIDGET_UNINSTALLATION_STARTED	3	The Widget uninstallation has started
WIDGET_UNINSTALLATION_COMPLETED	4	The Widget uninstallation has completed successfully
WIDGET_UNINSTALLATION_FAILED	5	The Widget uninstallation has failed
WIDGET_ERROR_STORAGE_AREA_FULL	10	The local storage device is full
WIDGET_ERROR_DOWNLOAD	11	The Widget cannot be downloaded
WIDGET_ERROR_INVALID_ZIP_ARCHIVE	12	The Widget package is corrupted or is an Invalid Zip Archive (as defined in W3C Widget Packaging and XML Configuration)
WIDGET_ERROR_INVALID_SIGNATURE	13	Widget's Signature Validation failed
WIDGET_ERROR_GENERIC	14	Other reason

Name	Value	Use
WIDGET_ERROR_SIZE_EXCEEDED	15	The Widget exceeded the maximum size for a single widget allowed by the platform, as defined in 9.1.
WIDGET_ERROR_PERMISSION_DENIED	16	The user and/or the OITF denied the installation or update of a Widget

7.2.2.3 Properties

function **onLowMemory()**

The function that is called when the OITF is running low on available memory for running DAE applications. The exact criteria determining when to generate such an event is implementation specific.

function **onApplicationLoaded**(Application appl)

The function that is called immediately prior to a **load** event being generated in the affected application. The specified function is called with one argument **appl**, which provides a reference to the affected application.

function **onApplicationUnloaded**(Application appl)

The function that is called immediately prior to an **unload** event being generated in the affected application. The specified function is called with one argument **appl**, which provides a reference to the affected application.

function **onApplicationLoadError**(Application appl)

The function that is called when the OITF fails to load either the file containing the initial HTML document of an application or an XML AIT file (e.g. due to an HTTP 404 error, an HTTP timeout, being unable to load the file from a DSM-CC object carousel or due to the file not being either an HTML file or a XML AIT file as appropriate). All properties of the **Application** object referred to by **appl** shall have the value **undefined** and calling any methods on that object shall fail.

function onWidgetInstallation(widgetDescriptor wd, Integer state, Integer reason)

The callback function that is called during the installation process of a Widget. The function is called with three arguments:

- widgetDescriptor wd – the widgetDescriptor for the installed Widget. Some attributes of this argument may not have been initialised and may be null when the function is called until the Widget is successfully installed.
- Integer state – the state of the installation; valid values are:
 - WIDGET_INSTALLATION_STARTED
 - WIDGET_INSTALLATION_COMPLETED
 - WIDGET_INSTALLATION_FAILED

as defined in 7.2.2.2.

- Integer reason: indicates the reason for installation failure. This is only valid if the value of the state argument is WIDGET_INSTALLATION_FAILED, otherwise this argument shall be null. Valid values for this field are:

- WIDGET_ERROR_STORAGE_AREA_FULL
- WIDGET_ERROR_DOWNLOAD
- WIDGET_ERROR_INVALID_ZIP_ARCHIVE
- WIDGET_ERROR_INVALID_SIGNATURE
- WIDGET_ERROR_GENERIC
- WIDGET_ERROR_SIZE_EXCEEDED
- WIDGET_ERROR_PERMISSION_DENIED

as defined in 7.2.2.2.

function onWidgetUninstallation(widgetDescriptor wd, Integer state)

The function that is called during the uninstallation process of a Widget. The function is called with two arguments, defined below:

- widgetDescriptor wd – the WidgetDescriptor of the Widget to be uninstalled.
- Integer state – the state of the installation; valid values are:
 - WIDGET_UNINSTALLATION_STARTED,
 - WIDGET_UNINSTALLATION_COMPLETED
 - WIDGET_UNINSTALLATION_FAILED

as defined in 7.2.2.2.

readonly widgetDescriptorCollection widgets

A collection of WidgetDescriptor objects for the Widgets currently installed on the OITF.

7.2.2.4 Methods

Integer <code>getApplicationVisualizationMode()</code>		
Description	Returns the current mode used by the OITF to visualize applications, whereby a return value:	
	1	corresponds to the application visualization mode as defined by item a) of 4.5.7, i.e. multiple applications visible simultaneously with DAE applications managing their own visibility
	2	corresponds to the application visualization mode as defined by item b) of 4.5.7, i.e. multiple applications visible simultaneously with OITF managing the size, position, visibility of applications
	3	corresponds to the application visualization mode as defined by item c) of 4.5.7, i.e. only a single application visible at any time.

Application <code>getOwnerApplication(Document document)</code>		
Description	Get the application that the specified document is part of. If the document is not part of an application, or the calling application does not have permission to access that application, this method will return <code>null</code> .	
Arguments	document	The document for which the application object should be obtained.

ApplicationCollection <code>getChildApplications(Application application)</code>		
Description	Get the applications that are children of the specified application.	
Arguments	application	The application whose children should be returned.

void <code>gc()</code>	
Description	Provide a hint to the execution environment that a garbage collection cycle should be initiated. The OITF is not required to act upon this hint.

void <code>installWidget(String uri)</code>		
Description	Attempts to install on the OITF a Widget located at the URI passed. If the Widget is stored on a remote server, it shall first be downloaded. This document does not specify where the OITF stores the Widget package, nor does it define what happens to the original package after the installation process has finished (regardless of whether it succeeded or failed).	
	When trying to install a Widget with an "id" that collides with the id of an already installed Widget (where the "id" is defined in 7.6.1 of W3C Widget Packaging and XML Configuration along with the extension defined in 11.2), the OITF should ask the user for confirmation before installing the Widget. The OITF should provide information about the conflict (e.g. the version numbers, if available) to allow the user to decide whether to proceed with the installation or to cancel it. If the user confirms the installation, then the new Widget shall replace the one already installed; any storage area associated with the replaced Widget shall be retained. Note that the user can also choose to downgrade a Widget, i.e. install an old version of the Widget to replace the installed, more recent, one.	
Arguments	uri	The resource locator in form of a URI, which points to a Widget package to be installed.

void <code>uninstallWidget(widgetDescriptor wd)</code>		
Description	Uninstalls a Widget. If this Widget is running, it will be stopped. Any storage areas associated with the uninstalled Widget shall be deleted.	
Arguments	wd	A <code>WidgetDescriptor</code> object for a Widget installed on the OITF.

7.2.2.5 Events

For the intrinsic events listed in the table below, a corresponding DOM event shall be generated in the following manner:

Intrinsic event	Corresponding DOM event	DOM Event properties
onLowMemory	LowMemory	Bubbles: No Cancellable: No Context Info: None
onApplicationLoaded	ApplicationLoaded	Bubbles: No Cancellable: No Context Info: app1
onApplicationUnloaded	ApplicationUnloaded	Bubbles: No Cancellable: No Context Info: app1
onApplicationLoadError	ApplicationLoadError	Bubbles: No Cancellable: No Context Info: app1
onWidgetInstallation	WidgetInstallation	Bubbles: No Cancellable: No Context: wd, state, reason
onWidgetUninstallation	WidgetUninstallation	Bubbles: No Cancellable: No Context: wd, state

The above DOM events are directly dispatched to the event target, and will not bubble nor capture. Applications should not rely on receiving the events listed above during the bubbling or the capturing phase. Applications that use DOM event handlers shall call the `addEventListener()` method on the `application/oipfApplicationManager` object. The third parameter of `addEventListener`, i.e. "useCapture", will be ignored.

7.2.3 Application class

7.2.3.1 General

The `Application` class is used to implement the characteristics of a DAE application.

If the document of an application is modified (or even replaced entirely), the `Application` object shall be retained. This means that the permission set granted when the application is created applies to all "edits" of the document or other pages in the application, until the application is destroyed.

7.2.3.2 Properties

readonly Boolean visible
true if the application is visible, false otherwise. The value of this property is not affected by the application's Z-index or position relative to other applications. Only calls to the <code>show()</code> and <code>hide()</code> methods will affect its value.

readonly Boolean active
true if the application is in the list of currently active applications, false otherwise (as defined in 4.4.9).

readonly stringCollection permissions
StringCollection object containing the names of the permissions granted to this application.

readonly Boolean isPrimaryReceiver
true if the application receives cross application events before any other application, false otherwise.

readonly window window
A strict subset of the DOM Window object representing the application. No symbols from the Window object are accessible through this property except the following: • void postMessage(any message, String targetOrigin)

readonly ApplicationPrivateData privateData
Access the current application's private data object. If an application attempts to access the privateData property of an Application object for a different application, the OITF shall throw an error as defined in 10.1.2.

<pre>function onApplicationActivated function onApplicationDeactivated function onApplicationShown function onApplicationHidden function onApplicationPrimaryReceiver function onApplicationNotPrimaryReceiver function onApplicationTopmost function onApplicationNotTopmost function onApplicationDestroyRequest function onApplicationHibernateRequest function onKeyPress function onKeyUp function onKeyDown</pre>
Each of these event handlers represents a DOM 0 event handler that corresponds to one of the events listed in 4.5.8 and 7.2.7.

7.2.3.3 Methods

void show()	
Description	If the application visualization mode as defined by method getApplicationVisualizationMode() in 7.2.2.4, is: 1: Make the application visible. 2: Make the application visible. Calling this method from the application itself may have no effect. 3: Request to make the application visible. This method only affects the visibility of an application. In the case where more than one application is visible, calls to this method will not affect the z-index of the application with respect to any other visible applications.

void hide()	
Description	If the application visualization mode as defined by method <code>getApplicationVisualizationMode()</code> in 7.2.2.4, is: 1: Make the application invisible. 2: Make the application invisible. Calling this method from the application itself may have no effect. 3: Request to make the application invisible. Calling this method has no effect on the lifecycle of the application. NOTE Broadcast independent applications should not call this method. Doing so may result in only the background being visible to the user

void activateInput(Boolean gainFocus)	
Description	Move the application to the front of the active applications list. If the application has been hidden using <code>Application.hide()</code>, this method does not cause the application to be shown. If the application visualization mode as defined by method <code>getApplicationVisualizationMode()</code> in 7.2.2.4, is: 1: The application's <code>Window</code> object shall be moved to the top of the stack of visible applications. In addition, the application's <code>Window</code> object shall gain input focus if argument <code>gainFocus</code> has value <code>true</code>. 2: The application's <code>Window</code> object shall be moved to the top of the stack of visible applications. In addition, the application's <code>Window</code> object shall gain input focus if argument <code>gainFocus</code> has value <code>true</code>. Calling this method from the application itself may have no effect. 3: Request to make the application's <code>Window</code> object visible. Once visible, the application shall be given input focus, irrespective of the value for argument <code>gainFocus</code>.

void deactivateInput()	
Description	Remove the application from the active applications list. This has no effect on the lifecycle of the application and may have no effect on the resources it uses. Applications which are not active will receive no cross-application events, unless their <code>Application</code> object is the target of the event (as for the events defined in 7.2.7). Applications may still be manipulated via their <code>Application</code> object or their DOM tree.

Application createApplication(String uri, Boolean createChild)		
Description	Create a new application and add it to the application tree. Calling this method does not automatically show the newly-created application. This call is asynchronous and may return before the new application is fully loaded. An <code>ApplicationLoaded</code> event will be targeted at the <code>Application</code> object when the new application has fully loaded. If the application cannot be created, this method shall return <code>null</code>.	
Arguments	uri	The URI of the first page of the application to be created or the <code>localURI</code> of a <code>Widget</code> as defined in 7.2.9.2.2.
	createChild	Flag indicating whether the new application is a child of the current application. A value of <code>true</code> indicates that the new application should be a child of the current application; a value of <code>false</code> indicates that it should be a sibling.

void destroyApplication()	
Description	Terminate the application, detach it from the application tree, and make any resources used available to other applications. When an application is terminated, any child applications shall also be terminated.

Application startWidget (widgetDescriptor wd, Boolean createChild)		
Description	Starts a Widget installed on the OITF. The behaviour of this method is equivalent to that of <code>Application.createApplication()</code>. The Widget is identified by its <code>WidgetDescriptor</code>. To get a list of the <code>WidgetDescriptor</code> objects for the installed Widgets one can check <code>ApplicationManager.widgets</code> property. If the Widget is already running or fails to start this call will return <code>null</code>.	
Arguments	wd	a <code>WidgetDescriptor</code> object for a Widget installed on the OITF.
	createChild	Flag indicating whether the new application is a child of the current application. A value of <code>true</code> indicates that the new application should be a child of the current application; a value of <code>false</code> indicates that it should be a sibling.

void stopWidget (widgetDescriptor wd)		
Description	Terminate a running Widget. The behaviour of this method is equivalent to that of <code>Application.destroyApplication()</code>. Calling this method will detach the Widget from the application tree, and make any resources used available to other applications. When a Widget is terminated, any child applications shall also be terminated.	
Arguments	wd	A <code>WidgetDescriptor</code> object for a Widget installed on the OITF.

7.2.4 The ApplicationCollection class

```
typedef Collection<Application> ApplicationCollection
```

The `ApplicationCollection` class represents a collection of `Application` objects. See Annex I for the definition of the collection template.

7.2.5 The ApplicationPrivateData class

7.2.5.1 Properties

readonly keyset keyset
The object representing the user input events sent to the DAE application.

readonly channel currentChannel
For a broadcast-related application, the value of the property contains the channel whose AIT is currently controlling the lifecycle of this application. If no channel is being presented, or if the application is not broadcast-related, the value of this property shall be <code>null</code> .

readonly Boolean wakeupApplication
The <code>wakeupApplication</code> property is set if there has been a <code>preparewakeupApplication()</code> request by that application.

readonly Boolean wakeupOITF
The <code>wakeupOITF</code> property is set if there has been a call to the <code>preparewakeupOITF()</code> method.

7.2.5.2 Methods

Integer getFreeMem()	
Description	Let application developer query information about the current memory available to the application. This is used to help during application development to find application memory leaks and possibly allow an application to make decisions related to its caching strategy (e.g. for images). Returns the available memory to the application or -1 if the information is not available. EXAMPLE: <pre>var app = appman.getOwnerApplication(window.document); debug("[APP] free mem = " + app.privateData.getFreeMem() + "\n");</pre>

Boolean preparewakeupApplication(String URI, String token, Date time)		
Description	The preparewakeupApplication() method allows the DAE application to set-up the OITF to wake-up at a specified time. The wake-up is limited to the OITF being in the PASSIVE_STANDBY state at the specified time. If the timer expires while the DAE application is in a different state, it is silently ignored. Only one wake-up is to be supported for a DAE application. If a previous wake-up request had been registered, it shall be overwritten. If the wake-up fails to be set up, this operation shall return false. Failure may be due to OITF expecting to change to an OFF power state, which would not allow the wake-up request to survive.	
Arguments	URI	The URI from which the content can be fetched.
	token	The token is a string which the application may retrieve with clearWakeupToken() .
	time	The time when the wake-up is to occur.

Boolean preparewakeupOITF(Date time)		
Description	The preparewakeupOITF() method allows the DAE application to set-up the OITF to wake-up at a specified time. The wake-up is limited to the OITF being in the PASSIVE_STANDBY or PASSIVE_STANDBY_HIBERNATE state at the specified time. If the timer expires while the DAE application is in a different state, it is silently ignored. Unlike preparewakeupApplication() this method applies to all the DAE applications and not limited to a single DAE application. If the wake-up fails to be set-up this operation shall return false. Failure may be due to OITF expecting to change to an OFF power state which would not allow the wake-up request to survive.	
Arguments	time	The time when the wake-up is to occur.

String clearwakeupToken()	
Description	The clearwakeupToken() method shall return the token set in preparewakeupApplication() method. The wake-up token should be cleared once it is read in order to limit usage to only when the DAE application starts up.

7.2.6 The Keyset class

7.2.6.1 General

The keyset object permits applications to define which key events they request to receive. There are two means of defining this. Common key events are represented by constants defined in this class which are combined in a bit-wise mask to identify a set of key events. Less common key events are not included in one of the defined constants and form part of an array.

The supported key events indicated through the capability mechanism in 9.3 shall be the same as the maximum set of key events available to the browser as indicated through this object.

The default set of key events available to broadcast-related applications shall be none. The default set of key events available to broadcast-independent or service provider related applications that do not call `keyset.setValue()` shall be all those indicated by the constants in this class that are supported by the OITF excluding those indicated by OTHER.

7.2.6.2 Constants

Constant name	Numeric Value	Use
RED	0x1	Used to identify the VK_RED key event.
GREEN	0x2	Used to identify the VK_GREEN key event.
YELLOW	0x4	Used to identify the VK_YELLOW key event.
BLUE	0x8	Used to identify the VK_BLUE key event.
NAVIGATION	0x10	Used to identify the VK_UP, VK_DOWN, VK_LEFT, VK_RIGHT, VK_ENTER and VK_BACK key events.
VCR	0x20	Used to identify the VK_PLAY, VK_PAUSE, VK_STOP, VK_NEXT, VK_PREV, VK_FAST_FWD, VK_REWIND, VK_PLAY_PAUSE key events.
SCROLL	0x40	Used to identify the VK_PAGE_UP and VK_PAGE_DOWN key events.
INFO	0x80	Used to identify the VK_INFO key event.
NUMERIC	0x100	Used to identify the number events, 0 to 9.
ALPHA	0x200	Used to identify all alphabetic events.
OTHER	0x400	Used to indicate key events not included in one of the other constants in this class.

7.2.6.3 Properties

readonly Integer value
The value of the keyset which this DAE application will receive.

readonly Integer otherKeys[]
If the OTHER bit in the value property is set, then this indicates those key events which are available to the browser which are not included in one of the constants defined in this class. If the OTHER bit in the value property is not set, then this property is meaningless.

readonly Integer maximumValue
In combination with maximumOtherKeys , this indicates the maximum set of key events which are available to the browser. When a bit in this maximumValue has value 0, the corresponding key events are never available to the browser.

readonly Integer maximumOtherKeys[]
If the OTHER bit in the maximumValue property is set then, in combination with maximumValue , this indicates the maximum set of key events which are available to the browser. For key events which are not included in one of the constants defined in this class, if they are not listed in this array then they are never available to the browser. If the OTHER bit in the value property is not set then this property is meaningless.

Boolean supportsPointer
Applications that have been designed to handle Mouse Events can express it by using this property. Applications shall set this property to true to indicate that they support a pointer-based interaction model, i.e. that they listen to and handle Mouse Events as included in the Web Standards TV profile IEC 62766-5-2. They shall set it to false otherwise. If not set, an OITF shall assume that the application does not support a pointer based interaction model. Based on the value of this property, an OITF may decide to enable or disable the rendering of a free-moving cursor. NOTE OITFs are not required to support a pointer based input device even though they are recommended to do so. If pointer based input devices are supported, this is expressed via the +POINTER UI Profile fragment as described in 9.2.

7.2.6.4 Methods

Integer setValue(Integer value, Integer otherKeys[])		
Description	Sets the value of the keyset which this DAE application requests to receive. Where more than one DAE application is running, the events delivered to the browser shall be the union of the events requested by all running DAE applications. Under these circumstances, applications may receive events which they have not requested to receive. The return value indicates which keys will be delivered to this DAE application encoded as bit-wise mask of the constants defined in this class.	
Arguments	value	The value is a number which is a bit-wise mask of the constants defined in this class. For example; <pre>myKeyset = myApplication.privateData.keyset; myKeyset.setValue(0x00000013); myKeyset.setValue(myKeyset.INFO myKeyset.NUMERIC);</pre>
	otherkeys	This parameter is optional. If the value parameter has the OTHER bit set, then it is used to indicate the key events that the application wishes to receive which are not represented by constants defined in this class.

String getKeyIcon(Integer code)		
Description	Return the URI of the icon representing the physical key or other mechanism that is used by the terminal to generate the key event for the given keycode passed. It shall return null if the key has no icon associated with it.	
Arguments	code	The VK_ constant for the key whose icon should be returned.

String getKeyLabel(Integer code)		
Description	Return the textual label representing the physical key or other mechanism that is used by the terminal to generate the key event for the given keycode passed. It shall return null if the key has no textual label associated with it.	
Arguments	code	The VK_ constant for the key whose textual label should be returned.

7.2.7 New DOM events for application support

New events have been created that are raised on the Application objects in the application tree. These are normal events, not cross-application events, and are used to indicate changes in the state of an application. They are specified in Table 6.

Table 6 – New DOM events for application support

Event	Description
ApplicationActivated	Issued when an application focus change occurs to inform the recipient of the event that the application is now focussed.
ApplicationDeactivated	Issued when an application focus change occurs to inform the recipient of the event that the application is now no longer focussed.
ApplicationShown	Issued when an application has become visible.
ApplicationHidden	Issued when an application has become hidden.
ApplicationPrimaryReceiver	This event is issued to indicate that the target is now at the front of the active application list.
ApplicationNotPrimaryReceiver	This event is issued to indicate that the target is no longer at the front of the active application list.
ApplicationTopmost	This event is issued to indicate that the target is now the topmost (i.e. it has the highest Z-index and is not obscured by any other visible applications, for OITFs where multiple applications are visible simultaneously).
ApplicationNotTopmost	This event is issued to indicate that the target is no longer at the topmost application. For OITFs where only one application is visible at a time, this event indicates that the application is no longer visible to the user.
ApplicationDestroyRequest	This event is issued to indicate that the target application is about to be terminated. It is not issued when an application calls <code>destroyApplication()</code> method for itself (i.e. to exit itself). Non-responsive applications should be forcibly terminated by the OITF, including the case where listeners for <code>ApplicationDestroyRequest</code> events do not return promptly. The determination of when an application is "non-responsive" is terminal-specific. If an application does not register a listener for this event and there is a need for the system to terminate the application, then the application shall be terminated immediately.
ApplicationHibernateRequest	This event is issued to indicate that the OITF is about to enter a hibernate mode. The OITF shall start a short watchdog timer (e.g. 2 s). During this period the application may take any actions (for example to store the currently viewed channel in case of an unsuccessful start-up).

These events do not bubble and cannot be cancelled. Each of these events has a corresponding DOM 0 event handler property on the Application object.

7.2.8 Examples

7.2.8.1 General

The examples in 7.2.8.2 and 7.2.8.3 illustrate some aspects of the application model.

7.2.8.2 Locating the application object

The `ApplicationManager` class provides the `getOwnerApplication()` method, which returns the document's owning application node:

```
// Assumes that the application/oipfApplicationManager object has the ID
// "applicationmanager"
var appMgr = document.getElementById("applicationmanager");
var self = appMgr.getOwnerApplication(window.document);
```

All other application functionality is available from this object.

7.2.8.3 Creating a new application

Creating a new application is a simple matter of creating a new `Application` object.


```
// Assumes that the application/oipfApplicationManager object has the ID
// "applicationmanager"
var appMgr = document.getElementById("applicationmanager");
var self = appMgr.getOwnerApplication(window.document);
var child = self.createApplication( url_of_application, true );
```

A typical requirement on an application is to only become visible once it has fully loaded. To do this, it can take advantage of load events. Here is an example from a clock application that wants to load an image to become the background of the clock, upon which it can write the text of the clock.

```
<script>
function loaded() {

    var screen = document.defaultView.screen;
    var clock = document.getElementById('clock');

    setup_clock( clock.width, clock.height );

    // Assumes that the application/oipfApplicationManager object has the ID
    // "applicationmanager"
    var appMgr = document.getElementById("applicationmanager");
    var self = appMgr.getOwnerApplication(window.document);
    self.show();
}
</script>

<style> * { margin: 0cm } </style>

<body onload="loaded()">
    
</body>
```

7.2.9 Widget APIs

7.2.9.1 General

Subclause 7.2.9 defines APIs an author can use to interact with Widgets installed on the OITF. Note that the Widget lifecycle is managed through the application manager as defined in the previous subclauses.

7.2.9.2 The WidgetDescriptor class

7.2.9.2.1 General

The widgetDescriptor class is used to implement the characteristics of a DAE Widget. It extends the Widget interface defined in 11.4 with the properties below.

7.2.9.2.2 Properties

readonly String localURI
The URI of the installed Widget. It can be used as an argument to <code>ApplicationManager.createApplication()</code> to run the Widget. The value of this property should not represent the actual location of the Widget on the OITF's local storage.
readonly String downloadURI
The URI of the location from where the Widget package was downloaded. This property shall match the URI used as argument of <code>createApplication()</code> or <code>installWidget()</code> when installing the Widget.
readonly StringCollection defaultIcon
A collection of URI strings for all the available default icons. Default icons are defined in W3C Widget Packaging and XML Configuration. This collection only contains URIs for the icons currently available in the Widget package.

readonly StringCollection customIcons
A collection of URI strings for all the custom icons of the current Widget. Custom icons are defined in W3C Widget Packaging and XML Configuration.

readonly Boolean running
This flag indicates the running state of the Widget.

7.2.9.2.3 Clarifications

The `WidgetDescriptor` class is used to identify an installed Widget regardless of whether it is running or not, and so some clarification on the attribute values defined for the Widget interfaces W3C Widget Interface is needed. The attributes height and width are defined in W3C Widget Interface on the "Widget instance's viewport". When the Widget is not running, those attributes shall take the value defined in the Widget Manifest (if any), otherwise they shall be null. When the Widget is running, these attributes shall adhere to what is defined in W3C Widget Interface.

7.2.9.3 The WidgetDescriptorCollection class

```
typedef Collection<WidgetDescriptor> WidgetDescriptorCollection
```

The `WidgetDescriptorCollection` class represents a collection of `WidgetDescriptor` objects. See Annex I for the definition of the collection template.

7.3 Configuration and setting APIs

7.3.1 General

Subclause 7.3 defines the interface to configuration and user settings information. Hardware configuration of the OITF is managed via an instance of the `LocalSystem` object. This provides access to hardware information and provides an entry point to configure the outputs and network interfaces of the OITF. Settings relating to the user interface and behaviour of the platform software are managed via an instance of the `Configuration` object.

Subclause 7.3 is subject to security control (see 10.1.4.8), and only applies if `<configurationChanges>` has value `true`.

7.3.2 The application/oipfConfiguration embedded object

7.3.2.1 General

The OITF shall implement the "application/oipfConfiguration" object as defined below. This object provides an interface to the configuration and user settings facilities within the OITF.

7.3.2.2 Properties

readonly Configuration configuration
Accesses the configuration object that sets defaults and shows system settings.

readonly LocalSystem localSystem
Accesses the object representing the platform hardware.

function onIpAddressChange(NetworkInterface item, String ipAddress)
The function that is called when the IP address of a network interface has changed. The specified function is called with two arguments <i>item</i> and <i>ipAddress</i> . The <i>ipAddress</i> may have the value <i>undefined</i> if a previously assigned address has been lost.

7.3.2.3 Events

For the intrinsic event "onIpAddressChange", a corresponding DOM event shall be generated in the following manner:

Intrinsic event	Corresponding DOM event	DOM Event properties
onIpAddressChange	IpAddressChange	Bubbles: No Cancellable: No Context Info: <i>item</i> , <i>ipAddress</i>

The above DOM event is directly dispatched to the event target, and will not bubble nor capture. Applications should not rely on receiving an *IpAddressChange* event during the bubbling or the capturing phase. Applications that use DOM event handlers shall call the *addEventListener()* method on the *application/oipfConfiguration* object. The third parameter of *addEventListener*, i.e. "useCapture", will be ignored.

7.3.3 The Configuration class

7.3.3.1 General

The Configuration object allows configuration items within the system to be read and modified. This includes settings such as audio and subtitle languages, display aspect ratios and other similar settings. Unlike the *LocalSystem* object, this is concerned with software- and application-related settings rather than hardware configuration and control.

7.3.3.2 Properties

String preferredAudioLanguage
A comma-separated set of languages to be used for audio playback, in order of preference. Each language shall be indicated by its ISO 639-2 language code as defined in ISO 639-2.
String preferredSubtitleLanguage
A comma-separated set of languages to be used for subtitle playback, in order of preference. The subtitle component (see 7.16.5.6) that matches the highest ordered language shall be activated (equivalent to the <i>selectComponent</i> method) and all other subtitle components shall be deactivated (equivalent to the <i>unselectComponent</i> method). Each language shall be indicated by its ISO 639-2 language code as defined in ISO 639-2 or as a wildcard specifier "***". If the wildcard is included, it shall be the last item in the set. If no subtitle component in the content matches a language in this property and the wildcard is included, then the first (lowest) subtitle component shall be selected.
String preferredUILanguage
A comma-separated set of languages to be used for the user interface of a service, in order of preference. Each language shall be indicated by its ISO 639-2 language code as defined in ISO 639-2. If present, the HTTP <i>Accept-Language</i> header shall contain the same languages as the <i>preferredUILanguage</i> property with the same order of preference. NOTE The order of preference in the <i>Accept-Language</i> header is indicated using the quality factor.

String <code>countryId</code>
An ISO-3166 three character country code identifying the country in which the receiver is deployed.

Integer <code>regionId</code>
An integer indicating the time zone within a country in which the receiver is deployed. A value of 0 shall represent the eastern-most time zone in the country, a value of 1 shall represent the next time zone to the west, and so on. Valid values are in the range 0 to 60.

Integer pvrPolicy	
The policy dictates what mechanism the system should use when storage space is exceeded.	
Valid values are shown in the table below.	
Value	Description
0	Indicates a recording management policy where no recordings are to be deleted.
1	Indicates a recording management policy where only watched recordings may be deleted.
2	Indicates a recording management policy where only recordings older than the specified threshold (given by the pvrSaveDays and pvrSaveEpisodes properties) may be deleted.

Integer <code>pvrSaveEpisodes</code>
When the <code>pvrPolicy</code> property is set to the value 2, this property indicates the minimum number of episodes that shall be saved for series-link recordings.

Integer <code>pvrSaveDays</code>
When the <code>pvrPolicy</code> property is set to the value 2, this property indicates the minimum save time (in days) for individual recordings. Only recordings older than the save time may be deleted.

Integer <code>pvrStartPadding</code>
The default padding (measured in seconds) to be added at the start of a recording.

Integer <code>pvrEndPadding</code>
The default padding (measured in seconds) to be added at the end of a recording.

Integer <code>preferredTimeShiftMode</code>
The time shift mode indicates the preferred mode of operation for support of timeshift playback in the video/broadcast object. Valid values are defined in the <code>timeShiftMode</code> property in 7.13.3.3. The default value is 0, timeshift is turned off.

7.3.3.3 Methods

String getText (String key)		
Description	Get the system text string that has been set for the specified key.	
Arguments	key	A key identifying the system text string to be retrieved.

void setText (String key, String value)										
Description	Set the system text string that has been set for the specified key. System text strings are used for automatically generated messages in certain cases, e.g. parental control messages.									
Arguments	key	The key for the text string to be set. Valid keys are:								
		<table><tr><th>Key</th><th>Description</th></tr><tr><td>no_title</td><td>Text string used as the title for programmes and channels where no guide information is available. Defaults to "No information"</td></tr><tr><td>no_synopsis</td><td>Text string used as the synopsis for programmes where no guide information is available. Defaults to "No further information available"</td></tr><tr><td>manual_recording</td><td>Text string used to identify a manual recording. Defaults to "Manual Recording"</td></tr></table>	Key	Description	no_title	Text string used as the title for programmes and channels where no guide information is available. Defaults to "No information"	no_synopsis	Text string used as the synopsis for programmes where no guide information is available. Defaults to "No further information available"	manual_recording	Text string used to identify a manual recording. Defaults to "Manual Recording"
		Key	Description							
		no_title	Text string used as the title for programmes and channels where no guide information is available. Defaults to "No information"							
	no_synopsis	Text string used as the synopsis for programmes where no guide information is available. Defaults to "No further information available"								
manual_recording	Text string used to identify a manual recording. Defaults to "Manual Recording"									
value	The new value for the system text string.									

7.3.4 The LocalSystem class

7.3.4.1 General

The LocalSystem object allows hardware settings related to the local device to be read and modified.

NOTE The standbyState property has been removed from this class.

7.3.4.2 Constants

The following values are defined for the standby state of the OITF:

Name	Value	Use
OFF	0	The OITF is in the off state and no power is consumed. This is the case of a power outage or if the OITF has the ability to be completely turned off. Scheduled recording is not expected to work.
ON	1	The OITF is in normal working mode with user interaction. The DAE applications may render any presentation graphically.
PASSIVE_STANDBY	2	The OITF is in the lowest possible power consumption state (meeting regulations and certifications). The OITF may support wake-up from a passive standby in order, for example, to perform a scheduled recording.
ACTIVE_STANDBY	3	The OITF is in an intermediate power consumption state. The output to the display shall be inactive. In this state DAE applications may continue to operate.
PASSIVE_STANDBY_HIBERNATE	4	The OITF is in the lowest possible power consumption state (meeting regulations and certifications). If the platform supports hibernate mode then the OITF stores all applications in volatile memory to allow for quick startup.
RESTART	5	The OITF shall restart and return to a ON state.
FACTORY_RESET	6	Restart the OITF and reset all settings and data to an initial/factory state. The exact settings and data to be reset are implementation dependant. The use of the this operation with the setPowerState method is subject to security control defined in 10.1.4.9.

The following values are defined for the startup URL of the OITF:

Name	Value	Use
STARTUP_URL_NONE	0	No startup URL is known.
STARTUP_URL_DHCP	1	The startup URL is derived from DHCP procedures.
STARTUP_URL_TR069	2	The startup URL is derived through TR-069 procedures.
STARTUP_URL_PRECONFIGURED	3	The startup URL is that which is configured through the OITF firmware.
STARTUP_URL_OTHER	9	The startup URL is obtained through other (perhaps non-standardized) procedures.

7.3.4.3 Properties

readonly String deviceID
Private OITF Identifier. This property shall take the value undefined except when accessed by applications meeting either of the following criteria: • The application is signalled in an SD&S service provider discovery record with an application usage of urn:oipf:cs:ApplicationUsageCS:2009:hni-igi where the SD&S service provider discovery record was obtained by the OITF through the procedure defined in 6.4.1.2 of IEC 62766-4-1:2017. • The URL of the application was discovered directly through the procedure defined in 6.4.1.2 of IEC 62766-4-1:2017. In these two cases, it shall take the same value as defined for the DHCP client identifier in DHCP option 61 in 13.2.2.1 of IEC 62766-4-1:2017.

readonly Boolean systemReady
Indicates whether the system has finished initialising. A value of true indicates that the system is ready.

readonly String vendorName
String identifying the vendor name of the device.

readonly String modelName
String identifying the model name of the device.

readonly String familyName
String identifying the name of the family that the device belongs to. Devices in a family differ only by details that do not impact the behaviour of the OITF aspect of the device, e.g. screen size, remote control, number of HDMI ports, size of hard disc. Family names are allocated by the vendor and the combination of vendorName and familyName should uniquely identify a family of devices. Different vendors may use the same familyName , although they are recommended to use conventions that avoid this.

readonly String softwareVersion
String identifying the version number of the platform firmware.

readonly String hardwareVersion
String identifying the version number of the platform hardware.

readonly String serialNumber
String containing the serial number of the platform hardware.

readonly Integer releaseVersion
Release version of the OIPF specification implemented by the OITF. For instance, if the OITF implements release 2 version "1.0", this property should be set to 2.

readonly Integer majorVersion
Major version of the OIPF specification implemented by the OITF. For instance, if the OITF implements release 2 version "2.0", this property should be set to 2.

readonly Integer minorVersion
Minor version of the OIPF specification implemented by the OITF. For instance, if the OITF implements release 2 version "2.0", this property should be set to 0.

readonly String oipfProfile
Profile of the OIPF specification implemented by the OITF. Values of this field are not defined in this part.

readonly Boolean pvrEnabled
Flag indicating whether the platform has PVR capability (local PVR). NOTE This property is deprecated in favour of the pvrSupport property.

readonly Boolean ciplusEnabled
Flag indicating whether the platform has CI+ capability.

readonly Integer powerState
The powerState property provides the DAE application the ability to determine the current state of the OITF. The property is limited to the ACTIVE_STANDBY or ON states.

readonly Integer previousPowerState
The previousPowerState property provides the DAE application the ability to retrieve the previous state.

readonly Integer timeCurrentPowerState
The time that the OITF entered the current power state. The time is represented in seconds since midnight (GMT) on 1/1/1970.

function onPowerStateChange(Integer powerState)
The function that is called when the power state has changed. The specified function is called with the argument powerState : Integer powerState – the new power state.

Integer volume

Get or set the overall system volume. Valid values for this property are in the range 0 – 100. The OITF shall store this setting persistently.

Boolean mute

Get or set the mute status of the default audio output(s). A value of **true** indicates that the default output(s) are currently muted.

readonly AVOutputCollection outputs

A collection of **AVOutput** objects representing the audio and video outputs of the platform. Applications may use these objects to configure and control the available outputs.

readonly NetworkInterfaceCollection networkInterfaces

A collection of **NetworkInterface** objects representing the available network interfaces.

readonly TunerCollection tuners

A collection of **Tuner** objects representing the physical tuners available in the OITF.

readonly Integer tvStandardsSupported

Get the TV standard(s) which are supported on the analogue video outputs.

This property can take one or more of the following values:

Value	Description
1	Indicates platform support for the NTSC TV standard.
2	Indicates platform support for the PAL-BGH TV standard.
4	Indicates platform support for the SECAM TV standard.
8	Indicates platform support for the PAL-M TV standard.
16	Indicates platform support for the PAL-N TV standard.

Values are stored as a bitfield.

readonly Integer tvStandard

Get the TV standard for which the analogue video outputs are currently configured.

This property can take one or more of the following values:

Value	Description
0	Indicates there are no analogue video outputs
1	Indicates platform support for the NTSC TV standard.
2	Indicates platform support for the PAL-BGH TV standard.
4	Indicates platform support for the SECAM TV standard.
8	Indicates platform support for the PAL-M TV standard.
16	Indicates platform support for the PAL-N TV standard.

readonly Integer pvrSupport	
Flag indicating the type of PVR support used by the application. This property may take zero or more of the following values:	
Value	Description
0	PVR functionality is not supported. This is the default value if <recording> as specified in 9.3.4 has value false .
1	PVR functionality is supported in the OITF. This is the default value if <recording> as specified in 9.3.4 has value true .
Values are stored as a bitfield.	

readonly StartupInformation startupInformation
Indicates any information used at startup time of the OITF.

function onStartupInfoChange (StartupInformation startupInfo)
The function that is called when any property in the startup information is changed.
The specified function is called with the argument startupInfo :
StartupInformation startupInfo – the new startup information.

7.3.4.4 Methods

Boolean setScreenSize (Integer width, Integer height)		
Description	Set the resolution of the graphics plane. If the specified resolution is not supported by the OITF, this method shall return false . Otherwise, this method shall return true .	
Arguments	width	The width of the display, in pixels.
	height	The height of the display, in pixels.

Boolean setTVStandard (Integer tvStandard)		
Description	Set the TV standard to be used on the analogue video outputs. Returns false if the requested mode cannot be set.	
Arguments	tvStandard	The TV standard to be set. Valid values are defined in the description of the tvStandard property in 7.3.4.3.

Integer setPvrSupport (Integer state)								
Description	Set the type of PVR support used by the application. The types of PVR supported by the receiver may not be supported by the application; in this case, the return value indicates the pvr support that has been set.							
Arguments	state	The type of PVR support desired by the application. More than one type of PVR functionality may be specified, allowing the receiver to automatically select the appropriate mechanism. Valid values are:						
		<table><tr><th>Value</th><th>Description</th></tr><tr><td>0</td><td>PVR functionality is not supported. This is the default value if <recording> as specified in 9.3.4 has value false.</td></tr><tr><td>1</td><td>PVR functionality is supported in the OITF. This is the default value if <recording> as specified in 9.3.4 has value true.</td></tr></table>	Value	Description	0	PVR functionality is not supported. This is the default value if <recording> as specified in 9.3.4 has value false .	1	PVR functionality is supported in the OITF. This is the default value if <recording> as specified in 9.3.4 has value true .
		Value	Description					
		0	PVR functionality is not supported. This is the default value if <recording> as specified in 9.3.4 has value false .					
1	PVR functionality is supported in the OITF. This is the default value if <recording> as specified in 9.3.4 has value true .							
Values are stored as a bitfield.								

Boolean setPowerState(Integer type)		
Description	The setPowerState() method allows the DAE application to modify the OITF state. The power state change may be restricted for some values of type, for example OFF, PASSIVE_STANDBY, RESTART and FACTORY_RESET. A call to setPowerState() with a restricted value of type shall return false.	
Arguments	type	The type values that may be specified are defined in 7.3.4.2.

Boolean setDigestCredentials(String protocol, String domain, String username, String password)		
Description	Set the credentials for the specified protocol to use for digest authentication negotiation for all subsequent requests to the specified domain. The credentials are persistently stored overwriting any previous set credentials. If domain is null, the provided credentials shall apply for all domains. Returns true if credentials are successfully set, false otherwise. If digest authentication is not supported for the specified protocol then return false. The valid values are the strings "http" and "https". Setting of Digest Credentials on the same protocol and domain shall update the username and password. If the credentials, when used, are incorrect then the behaviour shall be the same as any other time that stored credentials are incorrect, e.g. saved values from a user prompt. The credentials shall be used (if necessary) in all requests made by DAE applications. The credentials may be used in requests made by other components such as media players, DLNA clients, etc.	
Arguments	protocol	The protocol to apply the credentials.
	domain	The domain to which the credentials apply.
	username	The username to be used in the digest authentication.
	password	The password to be used in the digest authentication.

Boolean clearDigestCredentials(String protocol, String domain)		
Description	Clear any previously set digest credentials for the specified domain. If domain is null, all set credentials are cleared. Returns true if the digest credentials for the given protocol and domain were cleared or do not exist, or false if credentials failed to be cleared.	
Arguments	protocol	The protocol to apply the credentials. The value should be the same as one of those specified for the setDigestCredentials() method.
	domain	The domain to which the credentials apply.

Boolean hasDigestCredentials(String protocol, String domain)		
Description	Check if digest credentials are currently defined for the specified protocol and domain. Returns true if credentials have been set by a previous call to setDigestCredentials(), otherwise returns false.	
Arguments	protocol	The protocol to apply the credentials. The value should be the same as one of those specified for the setDigestCredentials() method.
	domain	The domain to which the credentials apply.

7.3.4.5 Events

For the intrinsic events listed in the table below, a corresponding DOM event shall be generated in the following manner:

Intrinsic event	Corresponding DOM event	DOM Event properties
onPowerStateChange	PowerStateChange	Bubbles: No Cancellable: No Context Info: powerState
onStartupInfoChange	StartupInfoChange	Bubbles: No Cancellable: No Context Info: startupInfo

The above DOM events are directly dispatched to the event target, and will not bubble nor capture. Applications should not rely on receiving the events listed above during the bubbling or the capturing phase. Applications that use DOM event handlers shall call the `addEventListener()` method on the `LocalSystem` object.

7.3.5 The NetworkInterface class

7.3.5.1 General

The `NetworkInterface` class represents a physical or logical network interface in the receiver.

7.3.5.2 Properties

readonly String ipAddress
The IP address of the network interface, in dotted-quad notation for IPv4 or colon-hexadecimal notation for IPv6. This property shall take the value <code>undefined</code> if no IP address has been assigned. The IP address may be link-local, private or global, depending on which address block it belongs to, as reserved by IANA.
readonly String macAddress
The colon-separated MAC address of the network interface.
readonly Boolean connected
Flag indicating whether the network interface is currently connected.
Boolean enabled
Flag indicating whether the network interface is enabled. Setting this property shall enable or disable the network interface.

7.3.6 The AVOutput class

7.3.6.1 General

The `AVOutput` class represents an audio or video output on the local platform.

7.3.6.2 Properties

readonly string **name**

The name of the output. Each output shall have a name that is unique on the local system. At least one of the outputs shall have the name "all" and shall represent all available outputs on the platform. The results of reading properties from the "all" AVOutput are implementation specific.

readonly string **type**

The type of the output. Valid values are "audio", "video", or "both".

Boolean **enabled**

Flag indicating whether the output is enabled. Setting this property shall enable or disable the output.

Boolean **subtitleEnabled**

Flag indicating whether the subtitles are enabled. The language of the displayed subtitles is determined by a combination of the value of the **Configuration.preferredSubtitleLanguage** property (see 7.3.3) and the subtitles available in the stream. For audio outputs, setting this property will have no effect.

String **videoMode**

Read or set the video format conversion mode, for which hardware support may be available on the device. Valid values are:

normal

stretch

zoom

The following table provides guidance as to the relationship between **videoMode**, **aspectRatio** (output) and the **aspectRatio** (input) of the AVVideoComponent class.

aspectRatio (input/output) value	videoMode value		
	Normal	Stretch	Zoom
16:9 input / 4:3 output	Black bars at top and bottom, all video visible	No black bars, picture stretched vertically	No black bars, picture clipped on left and right sides
4:3 input / 16:9 output	Black bars on left and right, all video visible	No black bars, picture stretched horizontally	No black bars, picture clipped top and bottom
4:3 input / 4:3 output	No change	No change	No change
16:9 input / 16:9 output	No change	No change	No change

The DAE application graphical layer is unaffected by the **videoMode**.

For audio-only outputs, setting this property shall have no effect.

String **digitalAudioMode**

Read or set the output mode for digital audio outputs for which hardware support may be available on the device. Valid values are shown below.

Value	Behaviour
ac3	Output AC-3 audio.
uncompressed	Output uncompressed PCM audio.

For video-only outputs, setting this property shall have no effect.

String audioRange	
Read or set the range for digital audio outputs for which hardware support may be available on the device. Valid values are shown below	
Value	Behaviour
normal	Use the normal audio range.
narrow	Use a narrow audio range.
wide	Use a wide audio range.
For video-only outputs, setting this property shall have no effect.	

String hdVideoFormat
Read or set the video format for HD and 3D video outputs for which hardware support may be available on the device. Valid values are:
480i
480p
576i
576p
720i
720p
1080i
1080p
720p_TaB
720p_SbS
1080i_SbS
1080p_TaB
1080p_SbS
For audio-only or standard-definition outputs, setting this property shall have no effect.

String tvAspectRatio
Indicates the output display aspect ratio of the display device connected to this output for which hardware support may be available on the device. Valid values are:
4:3
16:9
For audio-only outputs, setting this property shall have no effect.

readonly StringCollection supportedVideoModes
Read the video format conversion modes that may be used when displaying a 4:3 input video on a 16:9 output display or 16:9 input video on a 4:3 output display. The assumption is that the hardware supports conversion from either format and there is no distinction between the two. See the definition of the videoMode property for valid values.
For audio outputs, this property will have the value null .

readonly StringCollection supportedDigitalAudioModes

Read the supported output modes for digital audio outputs. See the definition of the `digitalAudioMode` property for valid values.

For video outputs, this property will have the value `null`.

readonly StringCollection supportedAudioRanges

Read the supported ranges for digital audio outputs. See the definition of the `audioRange` property for valid values.

For video outputs, this property will have the value `null`.

readonly StringCollection supportedHdVideoFormats

Read the supported HD and 3D video formats. See the definition of the `hdVideoFormat` property for valid values.

For audio outputs, this property will have the value `null`.

readonly StringCollection supportedAspectRatios

Read the supported TV aspect ratios. See the definition of the `tvAspectRatio` property for valid values.

For audio outputs, this property will have the value `null`.

readonly Integer current3DMode

Read whether the display is currently in a 2D or 3D mode. Return values are:

Value	Description
0	The display is in a 2D video mode
1	The display is in a 3D video mode

function on3DModeChange (Integer action)

This function is the DOM 0 event handler for events relating to actions carried out on an item in a content catalogue. The specified function is called with the following arguments:

- **Integer action** – The type of action that the event refers to. Valid values are:

Value	Description
0	The display changed from a 3D to a 2D video mode
1	The display changed from a 2D to a 3D video mode

7.3.6.3 Events

For the intrinsic events listed in the table below, a corresponding DOM event shall be generated, in the following manner:

Intrinsic event	Corresponding DOM event	DOM Event properties
<code>on3DModeChange</code>	<code>3DModeChange</code>	Bubbles: No Cancellable: No Context Info: <code>action</code>

The above DOM event is directly dispatched to the event target, and will not bubble nor capture. Applications should not rely on receiving an `IpAddressChange` event during the bubbling or the capturing phase. Applications that use DOM event handlers shall call the

`addEventListener()` method on the `application/oipfConfiguration` object. The third parameter of `addEventListener`, i.e. "useCapture", will be ignored.

7.3.7 The `NetworkInterfaceCollection` class

```
typedef Collection<NetworkInterface> NetworkInterfaceCollection
```

The `NetworkInterfaceCollection` class represents a collection of `NetworkInterface` objects. See Annex K for the definition of the collection template.

7.3.8 The `AVOutputCollection` class

```
typedef Collection<AVOutput> AVOutputCollection
```

The `AVOutputCollection` class represents a collection of `AVOutput` objects. See Annex I for the definition of the collection template.

7.3.9 The `TunerCollection` class

```
typedef Collection<Tuner> TunerCollection
```

The `TunerCollection` class represents a collection of `Tuner` objects. See Annex I for the definition of the collection template.

7.3.10 The `Tuner` class

7.3.10.1 General

A `Tuner` object represents the source of broadcast content provided through a physical tuner in the OITF. Each `Tuner` object is represented by a `<video_broadcast>` element in the capability description as defined in 9.3.2.

A `Tuner` object that is capable of tuning at the same time to multiple transponders shall have the *nrstreams* attribute of the `<video_broadcast>` element set to a value equal to the number of transponders.

A `Tuner` object that is capable of tuning to transponders of different types shall include all those types in the *types* attribute of the `<video_broadcast>` element.

NOTE An OITF may contain a physical tuner that has its capabilities split into multiple `Tuner` objects to fit the restrictions on the `<video_broadcast>` element outlined above and in 9.3.2.

7.3.10.2 Properties

readonly Integer <code>id</code>
A unique identifier of the tuner.

readonly String <code>name</code>
The name of the tuner as designated in OITF.

readonly IntegerCollection <code>idTypes</code>
Returns a collection of the types supported by the tuner. The types are according to the ID types in 7.13.12.2 under <code>Channel</code> object.

Boolean enableTuner
The property enables (true) and disables (false) the tuner. Reading the property provides the current state, enabled or disabled. Attempting to disable the tuner while the resource is in use has no effect and the tuner shall continue to be enabled. While disabled: - any external power feed (if applicable) shall be turned off; - the value of the signalInfo property is not defined; - the value of the lnbInfo property is not defined; - the tuner shall not be available for use by any JavaScript object (e.g. the video/broadcast object) or by the underlying OITF system (e.g. to perform a scheduled recording). Note the property enableTuner is available in order to re-enable the tuner and get access to the tuner again. The set value of the property shall persist after OITF restarts.

readonly SignalInfo signalInfo
The property returns a SignalInfo object with signal information for example signal strength.

readonly LNBInfo lnbInfo
The property returns a LNBInfo object with information regarding the LNB associated with the tuner.

readonly Integer frontEndPosition
Indicates the physical interface associated with the tuner.

Boolean powerOnExternal
The property turns on (true) and off (false) the power applied to the external interface of the tuner unless the tuner is disabled. Reading the property provides the current value, on or off. Attempting to modify the property while the resource is in use has no effect. The value of the property shall persist after OITF restarts. For DVB-S/S2, power is supplied to the LNB(s) and, if present, the DiSEqC switch. For DVB-T/T2, a supply +5V is supplied to the antenna with a built-in amplifier. Note that applying power may have adverse effects to the external equipment if it has its own power supply. It is a strong recommendation to indicate to the end user a possible adverse effect before using this method. For DVB-C/C2 there is no effect. Reading the property provides the current value.

7.3.11 The SignalInfo class

7.3.11.1 General

The **SignalInfo** object provides details on the signal strength of the tuner. If the tuner is not tuned to a transponder, the all properties shall have the value **undefined**.

7.3.11.2 Properties

readonly Number strength
Signal strength measured in dBm, for example -31,5dBm.

readonly Integer quality
Signal quality with range from 0 to 100. Calculation of quality is a function of ber and snr . The specification remains silent as to how the calculation is made.

readonly Integer ber
Bit error rate.

readonly Number snr
Signal to noise ratio (dB), for example 22,3dB.

readonly Boolean lock
True if the tuner is locked to a transponder.

7.3.12 The LNBInfo class

7.3.12.1 General

The LNBInfo object provides details on the LNB attached to a tuner. Setting any of the properties in this class results in an immediate update of the LNB configuration that is active for the associated tuner. The LNB configuration is stored persistently.

7.3.12.2 Constants

The following constants are defined in the LNBInfo class:

Name	Value	Use
DUAL_LO_FREQ_LNB	30	A universal LNB that has two local oscillator frequency settings available. The selection between the frequencies is done by the presence of a 22 kHz control signal.
SINGLE_LO_FREQ_LNB	31	Only a single local oscillator frequency is available in the LNB.

7.3.12.3 Properties

Integer lnbType
The type of LNB connected to the frontend. Valid values are listed in 7.3.12.2.

Number lnbLowFreq
The low or only, if a single local oscillator frequency LNB is used, LNB local oscillator frequency in MHz.

Number lnbHighFreq
If a dual local oscillator frequency LNB is used this is the high LNB local oscillator frequency in MHz. If a single local oscillator frequency LNB is used, this argument shall be set to 0.

Number crossoverFrequency
Indicates the frequency (in MHz) when to switch between the high- and low-band oscillator frequencies (lnbLowFreq and lnbHighFreq respectively).

Number lnbStartFrequency
Indicates the lowest frequency, in MHz, that the LNB can be used for.

Number <code>lnbStopFrequency</code>
Indicates the highest frequency, in MHz, that the LNB can be used for.

Number <code>orbitalPosition</code>
Indicates the orbital position of the satellite in degrees, negative value for west, positive value for east. For example, Astra 19,2 East would have orbitalPosition 19,2. Thor 0,8 West would have orbitalPosition -0,8. This property, if provided, will be used to select a tuner instance (when scanning and tuning). Setting any value which is not a valid orbital position (an absolute value greater than 180) indicates that the orbital position need not be considered when using the associated tuner.

7.3.13 The StartupInformation class

7.3.13.1 General

This class contains information pertaining to the startup characteristics and configuration of the OITF.

7.3.13.2 Properties

readonly Integer <code>urlSource</code>
The mechanism used to obtain the <code>url</code> property. Any of the <code>STARTUP_URL_*</code> values defined in 7.3.4.2 are valid.

readonly String <code>url</code>
The URL used at startup of the OITF. If the <code>urlSource</code> property is <code>STARTUP_URL_NONE</code> , then the value of this property shall be <code>NULL</code> . If the <code>urlSource</code> property is <code>STARTUP_URL_PRECONFIGURED</code> , then the value of this property shall be undefined.

7.4 Content download APIs

7.4.1 General

This subclause defines the content-on-demand download interfaces for both DRM-protected and non-DRM protected content.

An OITF and a DAE application which have indicated support for downloading content by providing value "true" for element `<download>` in their capability profile as specified in 9.3.5 shall adhere to the following requirements.

NOTE Annex B clarifies the purpose and the use of these interfaces in more detail.

7.4.2 The application/oipfDownloadTrigger embedded object

7.4.2.1 General

An OITF shall support a non-visual embedded object of type `application/oipfDownloadTrigger`, with the JavaScript API in 7.4.2.2 to enable passing a content-access descriptor to an underlying download manager using JavaScript.

The functionality as described in 7.4.2 is subject to the security model of Clause 10.

7.4.2.2 Methods

String registerDownload(String contentAccessDownloadDescriptor, Date downloadStart, Integer priority)		
Description	Send contentAccessDownloadDescriptor to the underlying download manager as a String formatted according to the content access download descriptor XML Schema as specified in Annex C. Returns a String value representing a unique identifier to identify the download, if the contentAccessDownloadDescriptor is valid and is accepted for triggering a download. If the OITF supports the application/oipfDownloadManager as specified in 7.4.4, this shall be the value of the "id" attribute of the associated Download object. Note that if the content access download descriptor contains multiple content items to be downloaded, the associated Download objects for each of these content items shall have the same value for the "id" value. The associated Download objects can be retrieved through the method getDownloads() as defined in 7.4.4.4. The OITF shall guarantee that download identifiers are unique in relation to recording identifiers and CODAsset identifiers. The method returns undefined if the contentAccessDownloadDescriptor is not accepted for triggering a download.	
Arguments	contentAccessDownloadDescriptor	String formatted according to the content access download descriptor XML Schema as specified in Annex C.
	downloadStart	Optional argument indicating the time at which the download should be started. If the argument is not included, or takes a value of null , then the download should start as soon as possible.
	priority	Optional argument indicating the relative priority of the download with respect to other downloads registered by the same organisation as the calling application. Higher values indicate a higher priority. If the argument is not included then a priority of 0 shall be assigned.

String registerDownloadURL(String URL, String contentType, Date downloadStart, Integer priority)		
Description	This method triggers the OITF to initiate a download of the content pointed to by the URL and the given content type. The contentType attribute shall reflect the expected type of content returned by the content server when connecting to the URL. The contentType can be used to evaluate if the content type is part of the list of accepted content types of the OITF. For example, if the OITF does not support content type "video/MP2T", then the registerDownloadURL method could return undefined to indicate this to the application in advance of the download. If contentType has value "application/vnd.oipf.ContentAccessDownload+xml", the method shall return a download identifier, after which the OITF shall immediately fetch the content access download descriptor, after which the same shall happen as if registerDownload() as defined in 4.7.4.2 with the given content access download descriptor as argument was called. The downloadStart argument only applies to the individual Download objects described by the content access download descriptor and shall not apply to the retrieval of the content access download descriptor itself. Note that if the content access download descriptor contains multiple content items to be downloaded, the associated Download objects for each of these content items shall have the same value for the "id" value. The associated Download objects can be retrieved through method getDownloads() as defined in 7.4.4.4. Returns a String value representing a unique identifier to identify the download, if the given arguments are acceptable by the OITF to trigger a download. If the OITF supports the application/oipfDownloadManager as specified in 7.4.4, this shall be the value of the "id" attribute of the associated Download object(s). The OITF shall guarantee that download identifiers are unique in relation to recording identifiers and CODASSET identifiers. The method returns undefined if the given arguments are not acceptable by the OITF to trigger a download.	
Arguments	URL	The URL from which the content can be fetched.
	contentType	The type of content referred to by the URL attribute. The contentType can be used to evaluate if the content type is part of the list of supported content types of the OITF.
	downloadStart	Optional argument indicating the time at which the download should be started. If the argument is not included, or takes a value of null then the download should start as soon as possible.
	priority	Optional argument indicating the relative priority of the download with respect to other downloads registered by the same organisation as the calling application. Higher values indicate a higher priority. If the argument is not included, then a priority of 0 shall be assigned.

Integer checkDownloadPossible(Integer sizeInBytes)		
Description	Checks whether a download of a given sizeInBytes would be possible at this moment in time. The value is application specific. For an application whose organization has a reservation, only the free space in the reservation shall be considered when making the check.	
	Possible return values are:	
	Value	Semantics
	0	Successful, i.e. the download could be successfully completed if it is started at this moment in time.
	1	Insufficient storage, i.e. the download could be started, but is unlikely to complete successfully, since insufficient storage capacity is available to fully store the content to be downloaded.
2	Storage not available, i.e. the download would fail, since the storage is currently unavailable, e.g. in the case of removable storage.	
Arguments	sizeInBytes	Integer value with the given size of the download in bytes.

7.4.3 Extensions to application/oipfDownloadTrigger

If an OITF has indicated support for both BCG metadata (i.e. by giving element <clientMetadata> value "true" and a "type" attribute with value "bcg"), and the download management APIs defined in 7.4.4 (i.e. by giving attribute "manageDownloads" of the <download> element a value unequal to "none") in the client capability description, then the following additional method shall be supported by the application/oipfDownloadTrigger object defined in 7.4.2.

The functionality as described in this subclause is subject to the security model of Clause 10.

String registerDownloadFromCRID (String CRID, String IMI, Date downloadStart, Integer priority)		
Description	Send (CRID, IMI) to underlying download manager. Returns a String value representing a unique identifier to identify the download if the (CRID, IMI) tuple is valid and is accepted for triggering a download. If the OITF supports the application/oipfDownloadManager as specified in 7.4.4, this shall be the value of the "id" attribute of the associated Download object(s), which corresponds to the CRID in this case. The OITF shall guarantee that download identifiers are unique in relation to recording identifiers and CODASSET identifiers. The method returns undefined if the given (CRID, IMI) tuple is not accepted for triggering a download. The values of the name, description, parentalRating and DRMControl properties shall be based on the metadata provided for the item matching that CRID.	
Arguments	CRID	The TV-Anytime Content reference ID that points to the general information about the item to download that does not change regardless of how the content is published or broadcast.
	IMI	The TV-Anytime Instance Metadata ID that points to the specific information related to the item to download, such as content location, usage rules (pay-per-view, etc.) and delivery parameters (e.g. video format).
	downloadStart	Optional argument indicating the time at which the download should be started. If the argument is not included, or takes a value of null then the download should start as soon as possible.
	priority	Optional argument indicating the relative priority of the download with respect to other downloads registered by the same organisation as the calling application. Higher values indicate a higher priority. If the argument is not included then a priority of 0 shall be assigned.

7.4.4 The application/oipfDownloadManager embedded object

7.4.4.1 General

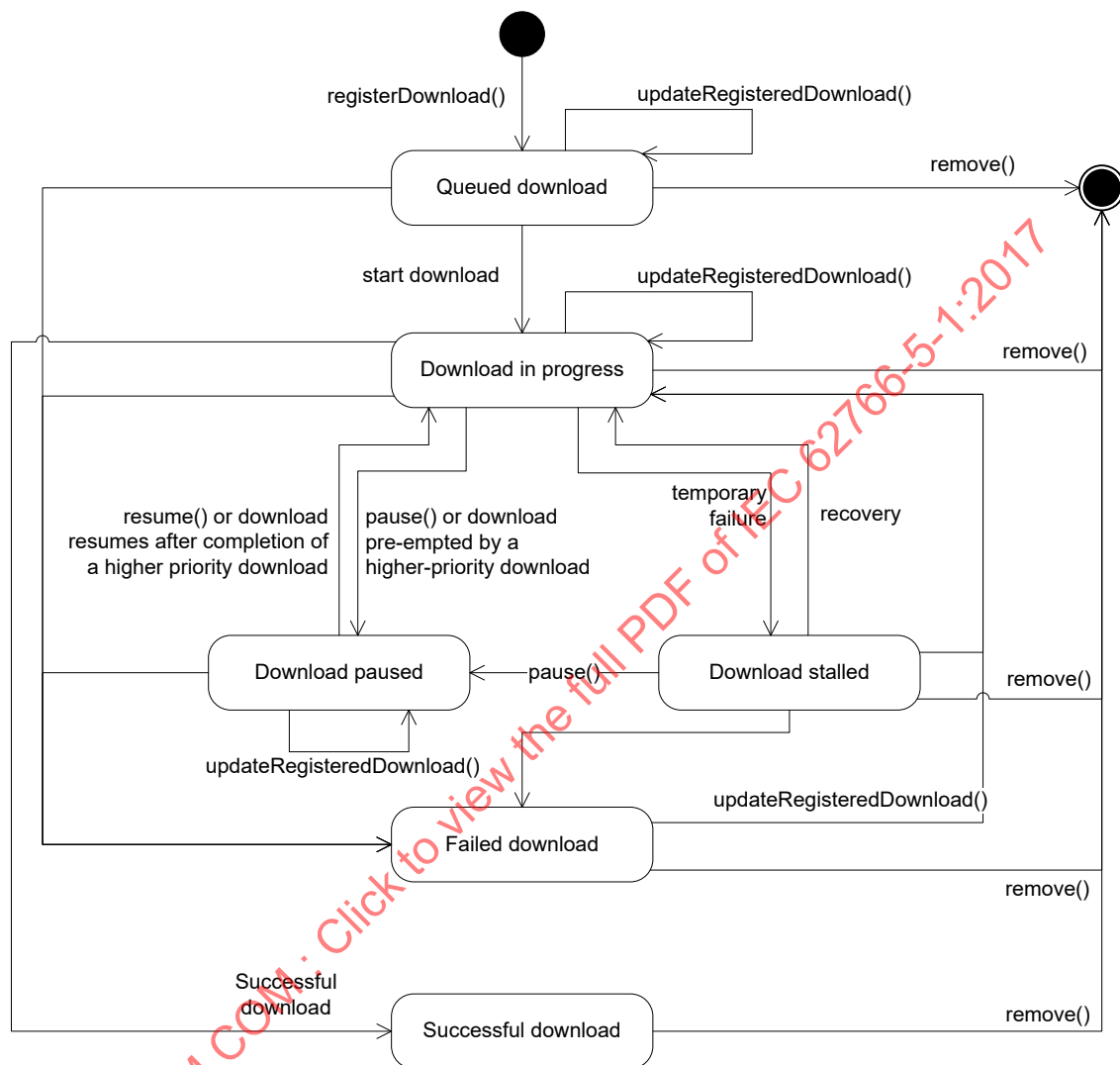
In a managed deployment, privileged applications may need access to the download management functionality in a CoD system. This access may be required to implement a UI to the download manager, to queue a download or to display the progress of a specific download. OITFs should support an "application/oipfDownloadManager" object with the following interface.

Clients supporting the download management APIs as specified in this subclause shall indicate this by adding the attribute "manageDownloads" to the <download> element with a value unequal to "none" in the client capability description as defined in 9.3.5.

The functionality as described in this subclause is subject to the security model of Clause 10.

7.4.4.2 State diagram for the application/oipfDownloadManager object

The following state machine (see Figure 12) provides an overview of the state changes that are possible in the download manager. The states reflect the changes signalled to applications via the onDownloadStateChange event handler.



IEC

NOTE Newly-registered downloads can preempt downloads that are currently in progress, if they have a higher priority than in-progress downloads. This can cause downloads to be paused or resumed without application intervention.

Figure 12 – State diagram for embedded application/oipfDownloadManager objects

7.4.4.3 Properties

function onDownloadStateChange(Download item, Integer state, Integer reason)																											
The function that is called when the status of a download has changed. The specified function is called with three arguments item, state and reason, which are defined as follows: - Download item – the Download object whose state has changed. - Integer state – the new state of the download. Valid values include: <table border="1"> <thead> <tr> <th>Status</th><th>Semantics</th></tr> </thead> <tbody> <tr> <td>1</td><td>The download has completed successfully.</td></tr> <tr> <td>2</td><td>The download is in progress.</td></tr> <tr> <td>4</td><td>The download has been paused (either by an application or automatically by the ONTF).</td></tr> <tr> <td>8</td><td>The download has failed.</td></tr> <tr> <td>16</td><td>The download has been queued but has not yet started.</td></tr> <tr> <td>32</td><td>The download has stalled due to a transient failure and the Download Manager is attempting to recuperate and re-establish the download.</td></tr> </tbody> </table> - Integer reason. Extended reason code. This is only valid if the value of the state argument is 8. <table border="1"> <thead> <tr> <th>Reason</th><th>Semantics</th></tr> </thead> <tbody> <tr> <td>0</td><td>The local storage device is full.</td></tr> <tr> <td>1</td><td>The item cannot be downloaded (e.g. because it has not been purchased).</td></tr> <tr> <td>2</td><td>The item is no longer available for download.</td></tr> <tr> <td>3</td><td>The item is invalid due to bad checksum or length.</td></tr> <tr> <td>4</td><td>Other reason.</td></tr> </tbody> </table> If no error has occurred, this argument shall take the value undefined.		Status	Semantics	1	The download has completed successfully.	2	The download is in progress.	4	The download has been paused (either by an application or automatically by the ONTF).	8	The download has failed.	16	The download has been queued but has not yet started.	32	The download has stalled due to a transient failure and the Download Manager is attempting to recuperate and re-establish the download.	Reason	Semantics	0	The local storage device is full.	1	The item cannot be downloaded (e.g. because it has not been purchased).	2	The item is no longer available for download.	3	The item is invalid due to bad checksum or length.	4	Other reason.
Status	Semantics																										
1	The download has completed successfully.																										
2	The download is in progress.																										
4	The download has been paused (either by an application or automatically by the ONTF).																										
8	The download has failed.																										
16	The download has been queued but has not yet started.																										
32	The download has stalled due to a transient failure and the Download Manager is attempting to recuperate and re-establish the download.																										
Reason	Semantics																										
0	The local storage device is full.																										
1	The item cannot be downloaded (e.g. because it has not been purchased).																										
2	The item is no longer available for download.																										
3	The item is invalid due to bad checksum or length.																										
4	Other reason.																										

readonly DiscInfo discInfo
Get information about the status of the local storage device. The DiscInfo class is defined in 7.16.4.

readonly Integer hasReserved
Returns the size (in megabytes) of any current reservation for the applications from the same organisation as the calling application, otherwise –1.

readonly Integer allocated
Returns the size (in megabytes) of any current reservation for the applications from the same organisation as the calling application, otherwise –1.

7.4.4.4 Methods

Boolean pause(Download download)		
Description	Pause an in-progress, queued or stalled download and return true. For in-progress downloads, more data shall not be downloaded until the download is resumed. The HTTP request and TCP socket are interrupted and closed. For completed or failed downloads, this operation shall return false.	
Arguments	download	The download to be paused.

Boolean resume (Download download)		
Description	Resume a paused download. If the download is not paused, this operation shall return false .	
Arguments	download	The download to be resumed.

Boolean remove (Download download)		
Description	Remove the download and any data and media content associated with it and return true. Return false if the download attribute does not refer to a valid download. As a side effect of this method, all properties on download shall be set to undefined. Any method calls subsequently performed by an application which pass download as an argument shall return false. If an A/V Control object is referring to the indicated download for playback, then the state of the A/V Control object shall be automatically changed to state 6 (the error state).	
Arguments	download	The download to be deleted.

DownloadCollection getDownloads (String id)		
Description	Returns a collection of downloads, for which the value of the Download.id property corresponds to the given id parameter. The downloads returned in the collection shall be filtered according to the value of the manageDownloads attribute of the <download> element in the OITF's capability description (i.e. from the same application, same domain or from all applications). For downloads initiated from registerDownloadURL() with a contentType value "application/vnd.oipf.ContentAccessDownload+xml" shall return null until the content access download descriptor has been retrieved and parsed. If the value of id is null, it returns all downloads for the scope indicated by the manageDownloads attribute.	
Arguments	id	Optional argument identifying the downloads to be retrieved. If present and not null , this is an identifier corresponding to the " id " attribute of zero or more Download objects. If the value of id is null , or the argument is not included, all downloads for the scope indicated by the manageDownloads attribute in the capability description are returned.

IECNORM.COM : Click to view the full PDF of IEC 62766-5-1:2017

DownloadCollection createFilteredList (Boolean currentDomain, Integer states)										
Description	Create a filtered list of downloads. Returns a subset of downloads that are managed by the receiver.									
	The currentDomain flag indicates whether downloads from FQDNs other than the current page are included in the returned collection. This flag may be set to one of three values:									
	<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>true</td><td>The download is added if and only if it was initiated from the FQDN of the calling document. If the application has the permission <i>permission_downloadmanager</i> (see 10.1.5), only downloads initiated by the calling application shall be added.</td></tr><tr><td>false</td><td>The download is added if and only if it was not initiated from the FQDN of the calling document. If the application does not have the permission <i>permission_downloadmanager_all</i> (see 10.1.5), the OITF shall return an empty collection.</td></tr><tr><td>undefined</td><td>The download is added regardless of the domain that the download was initiated from. If the application has the permission <i>permission_downloadmanager</i> (see 10.1.5), only downloads initiated by the calling application shall be added. If the application has the permission <i>permission_downloadmanager_samedomain</i> (see 10.1.5), only downloads initiated by applications from the same FQDN shall be added.</td></tr></table>	Value	Meaning	true	The download is added if and only if it was initiated from the FQDN of the calling document. If the application has the permission <i>permission_downloadmanager</i> (see 10.1.5), only downloads initiated by the calling application shall be added.	false	The download is added if and only if it was not initiated from the FQDN of the calling document. If the application does not have the permission <i>permission_downloadmanager_all</i> (see 10.1.5), the OITF shall return an empty collection.	undefined	The download is added regardless of the domain that the download was initiated from. If the application has the permission <i>permission_downloadmanager</i> (see 10.1.5), only downloads initiated by the calling application shall be added. If the application has the permission <i>permission_downloadmanager_samedomain</i> (see 10.1.5), only downloads initiated by applications from the same FQDN shall be added.	
	Value	Meaning								
	true	The download is added if and only if it was initiated from the FQDN of the calling document. If the application has the permission <i>permission_downloadmanager</i> (see 10.1.5), only downloads initiated by the calling application shall be added.								
false	The download is added if and only if it was not initiated from the FQDN of the calling document. If the application does not have the permission <i>permission_downloadmanager_all</i> (see 10.1.5), the OITF shall return an empty collection.									
undefined	The download is added regardless of the domain that the download was initiated from. If the application has the permission <i>permission_downloadmanager</i> (see 10.1.5), only downloads initiated by the calling application shall be added. If the application has the permission <i>permission_downloadmanager_samedomain</i> (see 10.1.5), only downloads initiated by applications from the same FQDN shall be added.									
The states flag indicates which state(s) of downloads that should be included in the list. The value of this flag is the arithmetic sum of one or more possible values of the state property of the Download object; only downloads whose state matches one of the values included in this sum are included in the returned collection.										
Arguments	currentDomain	Flag indicating whether downloads from other domains shall be added to the list.								
	states	Indicates that states of downloads that should be included in the returned list.								

Boolean updateRegisteredDownload (Download download, string newURL)		
Description	The method updateRegisteredDownload() provides a way to update the URL to be used for a download. The OITF shall use the new URL for any future retrieval.	
	If the download is already in progress or paused (indicated by a state property value of 4), it shall be stopped. The download shall continue from the last byte received during the previous download.	
	If the state property of the download argument has the value 8 (download failed) or 32 (download stalled), then the OITF shall resume the download from the last byte received during the previous download but using the new URL.	
	If the state property of the download argument has the value 16 (download not started), no further action is taken until the download is started or resumed.	
	If the state property of the download argument has the value 1 (download completed), then this method shall return false . Otherwise it shall return true .	
Arguments	download	The download object to be updated.
	newURL	The new URL from which the content can be retrieved.

Integer reserve (Integer bytes)		
Description	Requests the reservation of space for downloads to be made by applications from the same organization as the calling application. Reservation is optional for applications to use and applications may use the Download API without reserving space in advance. If there is already a reservation for the calling application's organization, this requests adjusting the reservation to the new value. If a call to reserve shrinks an already existing reservation, then the application should ensure that the new size is sufficient for all the completed, in progress and requested downloads by applications from the calling application's organization. The OITF shall refuse to shrink the reservation if this has not been done. This document intentionally does not define the criteria that are used in deciding whether or not to make or adjust a reservation. Either the OITF or the end-user may cancel a reservation completely, shrink one to recover some or all of the free space in the reservation or expand it. This document intentionally does not define circumstances when this may happen. If a request to reserve space is granted to an organisation then applications from that organisation shall have the reserved space available to them for downloads. Attempts to use more than the reserved space shall fail. A reservation is cancelled by calling this method with size zero. Returns one of the RESERVE_ constants defined in 7.4.4.6 below.	
Arguments	bytes	The number of bytes to reserve.

7.4.4.5 Events

For the intrinsic event "onDownloadStateChange", a corresponding DOM event shall be generated, in the following manner:

Intrinsic event	Corresponding DOM event	DOM Event properties
onDownloadStateChange	DownloadStateChange	Bubbles: No Cancellable: No Context Info: item, state , reason

The above DOM event is directly dispatched to the event target, and will not bubble nor capture. Applications should not rely on receiving a DownloadStateChange event during the bubbling or the capturing phase. Applications that use DOM event handlers shall call the `addEventListener()` method on the application/oipfDownloadManager object. The third parameter of `addEventListener`, i.e. "useCapture", will be ignored.

7.4.4.6 Constants

Name	Use
RESERVE_OK	Reservation succeeded
RESERVE_NEVER	Reservations by the calling application will never succeed.
RESERVE_USER_INTERVENTION_REQD	Reservation failed and user intervention is required. This result shall only be returned by OITFs that include a platform provided UI enabling end-users to manage storage. RESERVE_TOO_LARGE shall be returned in preference to this if both apply
RESERVE_USER_DECLINED	Reservation failed as the user was asked to approve this request and declined.
RESERVE_TOO_LARGE	Reservation failed as the size requested was larger than what is permitted by the OITF.□
RESERVE_SMALLER_THAN_USED	Request to shrink an already existing reservation failed as the requested size is smaller than the space currently used from the reservation.
RESERVE_UNKNOWN	Reservation failed for another reason.

readonly Integer currentBitrate
The bitrate (in bits per second) at which the download is currently transferred. This value is non-zero only when the Download object is in state 2 (in progress). If this is unknown, the value of this property shall be undefined .
String name
The name of the download or undefined if this information is not present. In case the download is triggered through a content access download descriptor, this corresponds to the value for the <Title> element in the content access download descriptor. If the content access download descriptor is not specified then the property may be set by the origin site. Note that the property may only be set by the site that initiated the download. The DAE application may store data related to the Download. The OITF shall support a minimum of 200 bytes for the property. If DAE application attempts to store a string larger than the available size the OITF shall set the property to NULL. The maximum length of the property value is implementation dependent.
readonly String id
The ID of the download as determined by the OITF.
readonly String uri
A uri identifying the content item in local storage according to IETF RFC 3986. The format of the URI is outside the scope of this document except that; - the scheme shall not be one that is included in this document - the URI shall not include a fragment
readonly String contentURL
The URL the content is being fetched from, or undefined if this information is not available.
String description
A description of the download or undefined if this information is not present. In case the download is triggered through a content access download descriptor, this corresponds to the value for the <Synopsis> element in the content access download descriptor, or undefined if this element is not present. If the content access download descriptor is not specified the property may be set by the origin site. Note that the property may only be set by the site that initiated the download. The DAE application may store data related to the Download. The OITF shall support a minimum of 2 000 bytes for the property. If DAE application attempts to store a string larger than the available size, the OITF shall set the property to NULL. The maximum length of the property value is implementation dependent.
readonly ParentalRatingCollection parentalRatings
The parental rating collection related to the downloaded content item, or undefined if this information is not present. In case the download is triggered through a content access download descriptor, this corresponds to the value for the <ParentalRating> element in the content access download descriptor, or undefined if this element is not present.
readonly DRMControlInfoCollection drmControl
The DRMControlInformation object corresponding to the DRM Control information of the downloaded content item, or undefined if this information is not present. In case the download is triggered through a content access download descriptor, this corresponds to the value for the <DRMControlInformation> element associated with the same DRMSystemID of the selected <ContentURL>, or is undefined if this information is not present. The related DRMControlInformation object is defined in 7.4.7.

readonly Date startTime

The time that the download is scheduled to start (in the case of scheduled downloads) or **undefined** if no start time was set.

readonly Integer timeElapsed

The time (in seconds) that has elapsed since the download of the item was started. The elapsed time shall be based on the time spent in the in-progress and stalled download states. This shall not include any time the item spent queued for download.

readonly Integer timeRemaining

The estimated time remaining (in seconds) for the download to complete. The estimated time shall be based on the time spent in the in-progress and stalled download states. The estimate shall not include any time the item spent queued for download or paused. If an estimate cannot be calculated, the value of this property shall be **undefined**.

readonly String transferType

In the case where the download was triggered through a content access download descriptor, this is the value of property **TransferType** of the selected **<ContentURL>**. In the case where the download was not triggered through a content access download descriptor, the OITF is responsible for returning either the value "playable_download" or "full_download", based on criteria defined by the OITF.

readonly String originSite

In the case where the download was triggered through a content access download descriptor, this is the value of element **<OriginSite>**. In case the download was not triggered through a content access download descriptor, this is the FQDN of the site that initiated the download.

readonly String originSiteName

In case the download is triggered through a content access download descriptor, this is the value of element **<OriginSiteName>**, or **undefined** if this information is not present. In the case where the download is not triggered through a content access download descriptor, this property is **undefined**.

String contentID

A unique identification of the content item relative to originSite. If the download is triggered through a content access download descriptor, and a **<ContentID>** element has been defined for the given content item, this is the value of element **<ContentID>**. If the download is started using **registerDownloadFromCRID()**, this is the TV Anytime CRID. This property shall take the value **undefined** if no content ID is available.

If the content access download descriptor is not specified the property may be set by the originSite. Note that the property may only be set by the site that initiated the download. The DAE application may store data related to the Download. The OITF shall support a minimum of 2 000 bytes for the property. If DAE application attempts to store a string larger than the available size, the OITF shall set the property to null. The maximum length of the property value is implementation dependent.

readonly String iconURL

The URL of an image that provides a visual representation of the item that is being downloaded. In the case where the download was triggered a content access download descriptor, this is the value of element **<IconURL>**, or **undefined** if this element is not present. In the case where the download was not triggered through a content access descriptor document, this property is **undefined**.

readonly Document metadata

For downloads registered through a content access download descriptor, this function shall return the contents of the content access download descriptor as an XML **Document** object using the syntax as defined in Clause C.1 without using any namespace definitions.

For downloads registered using a URL, the value of this property shall be null.

readonly Integer priority

The relative priority of the download with respect to other downloads registered by that application. Higher values indicate a higher priority.

readonly Boolean suspendedByTerminal

Flag indicating whether the download has been paused automatically by the OITF, either because the download has been pre-empted by higher priority downloads or because the number of simultaneous downloads supported by the OITF has been exceeded.

7.4.6 The DownloadCollection class

```
typedef Collection<Download> DownloadCollection
```

The DownloadCollection class represents a collection of Download objects. See Annex I for the definition of the collection template.

7.4.7 The DRMControlInformation class**7.4.7.1 General**

A DRMControlInformation object represents the DRM Control information structure defined in 4.4.3 of IEC 62766-3:2016.

7.4.7.2 Properties**readonly String drmType**

URN containing the decimal number format of the DVB CASystemID, prefixed with the string "urn:dvb:casystemid:". For example, the hexadecimal value 0x4AF4 is assigned as the CASystemID for Marlin by DVB, and so for Marlin the value of this property would be "urn:dvb:casystemid:19188".

readonly String rightsIssuerURL

A URL used by OITF to obtain rights for this content item.

readonly String silentRightsURL

A URL used by OITF to obtain rights silently, e.g. a Marlin Action Token.

readonly String drmContentID

DRM Content ID for CoD or scheduled content item, e.g. the Marlin Content ID.

readonly String previewRightsURL

A URL used by OITF to obtain rights silently for preview of this content item, e.g. a Marlin Action Token.

readonly String drmPrivateData
Private data for the DRM scheme indicated in drmType to be applied for this content item. Private DRM Data is actually structured as an XML document whose schema is specific to the considered DRM system. One example is Marlin DRM private data schema defined in IEC 62766-7.

readonly Boolean doNotRecord
A flag indicating whether this content item is recordable or not.

readonly Boolean doNotTimeShift
A flag indicating if this content item is allowed for time shift play back.

7.4.8 The DRMControlInfoCollection class

```
typedef Collection<DRMControlInformation> DRMControlInfoCollection
```

The `DRMControlInfoCollection` class represents a collection of `DRMControlInformation` objects. See Annex I for the definition of the collection template.

7.5 Content on demand metadata APIs

7.5.1 General

Subclause 7.5 shall apply for OITFs that have indicated `<clientMetadata>` with value "true" and a "type" attribute with value "bcg" in the capability description and may apply for OITFs that have indicated `<clientMetadata>` with value "true" and a "type" attribute with value "dvb-si".

7.5.2 The application/oipfCodManager embedded object

7.5.2.1 General

OITFs that have indicated `<clientMetadata>` with value "true" and a "type" attribute with value "bcg" shall implement an "application/oipfCodManager" embedded object with the following interface.

Content is organised into catalogues, where each catalogue contains a hierarchy of folders that are used to organise individual content items. The structure of the catalogue shall be determined by the server managing that catalogue and shall be reflected in the structure of the metadata passed to the OITF.

The three types of content in a CoD catalogue are:

- Assets, represented by the `CODAsset` class. A `CODAsset` is a user-level description of a piece of CoD content, and so it is more concerned with information such as the price, rental period, description and parental rating rather than detailed technical information about the asset such as encoding format. A CoD asset may represent a single movie, or a bundle of movies offered for a single price.
- Folders, represented by the `CODFolder` class.
- Services represented by the `CODService` class. `CODService` objects are a specific type of container representing subscription VoD (SVOD) services, where users purchase a group of assets which may change over time rather than a single movie or TV show.

The `CODAsset`, `CODFolder` and `CODService` classes define a type property that allows these classes to be distinguished by applications. For each class, this property shall take the value defined below:

Class	Value
CODFolder	0
CODAsset	1
CODService	2

This document defines the mapping between the CoD API and BCG metadata. Mappings between the CoD API and other CoD metadata sources are not specified in this document.

7.5.2.2 Properties

readonly ContentCatalogueCollection catalogues
A collection of all available CoD catalogues, as listed in an SD&S BCG Discovery record.

function onContentCatalogueEvent(Integer action)
This function is the DOM 0 event handler for events relating to changes in a content catalogue collection. The specified function is called with the argument action: Integer action – The type of event. For current versions of the specification, this property shall always have the value 0 to indicate a change in the list of available catalogues.

function **onContentAction**(Integer action, Integer result, Object item, ContentCatalogue catalogue)

This function is the DOM 0 event handler for events relating to actions carried out on an item in a content catalogue. The specified function is called with the following arguments:

- Integer action – The type of action that the event refers to. Valid values are:

Value	Description
0	An operation to browse a content collection (e.g. getting a page from the collection).
1	Indicates that more information is available about this item (e.g. that more information has been retrieved from the server).
- Integer result – The result of the action. Valid values are:

Value	Description
0	The operation succeeded.
1	The item no longer exists in the catalogue.
2	The server has not responded in the timeout period.
3	Communication with the server has been interrupted.
- Object item – The item in the catalogue that the event refers to.
- ContentCatalogue catalogue – The parent catalogue of the affected object.

7.5.2.3 Events

For the intrinsic events listed in the table below, a corresponding DOM event shall be generated in the following manner:

Intrinsic event	Corresponding DOM event	DOM Event properties
onContentCatalogueEvent	ContentCatalogueEvent	Bubbles: No Cancellable: No Context Info: action
onContentAction	ContentAction	Bubbles: No Cancellable: No Context Info: action, result, item, catalogue

The above DOM events are directly dispatched to the event target, and will not bubble nor capture. Applications should not rely on receiving the events listed above during the bubbling or the capturing phase. Applications that use DOM event handlers shall call the `addEventListener()` method on the `LocalSystem` object. The third parameter of `addEventListener`, i.e. "useCapture", will be ignored.

7.5.3 The ContentCatalogueCollection class

```
typedef Collection<ContentCatalogue> ContentCatalogueCollection
```

The `ContentCatalogueCollection` class represents a collection of `ContentCatalogue` objects. See Annex I for the definition of the collection template.

7.5.4 The ContentCatalogue class

7.5.4.1 General

A `ContentCatalogue` represents a content catalogue for a content on demand service.

To receive events relating to operations on items in a catalogue, applications may add listeners for "ContentAction" events to the `application/oipfCodManager` object.

7.5.4.2 Properties

readonly String name
The name of the content catalogue that should be displayed to the user. The value of this property is given by the Name element in the catalogue's BCG discovery record.
readonly CODFolder rootFolder
The root folder of the content catalogue.

7.5.4.3 Methods

CODFolder getPurchaseHistory()
Description Get the list of items that have been purchased from the catalogue by the current user, including currently active rentals. Items in this list will be derived from children of the BCG <code>UserActionList</code> element which have an <code>ActionType</code> of <code>buy</code>. If the <code>ActionList</code> element is not present, this method shall return <code>null</code>.

7.5.5 The ContentCatalogueEvent class

This subclause is intentionally left empty.

7.5.6 The CODFolder class

7.5.6.1 General

CODFolder represents a folder in a CoD catalogue. Folders may contain other folders, and an asset may be present in more than one folder.

Because a content list may contain a large number of items, the contents of the list are made available on demand using a paging model. Applications may request the contents of the list in 'pages' of an arbitrary size. The data shall be fetched from the appropriate source, and applications shall be notified when the data is available.

Each folder is described by a GroupInformation element in the BCG Group Information Table.

7.5.6.2 Properties

readonly Integer type
The type of the item, used to distinguish between the types of objects that may be contained in a folder in a CoD catalogue. This shall always have the value 0 for folders.
readonly String uri
The URI used to refer to the folder. The value of this property is given by the GroupId attribute of the GroupInformation element representing this folder.
readonly String name
The name of the folder. The value of this property is given by the Title element that is a descendant of the GroupInformation element representing this folder.
readonly String description
A description of the folder, for display to an end user. The value of this property is given by the Synopsis element that is a descendant of the GroupInformation element representing this folder.
readonly String thumbnailUri
The URI of an image associated with this folder. For assets whose BCG description contains a RelatedMaterial element indicating a relationship of Promotional Still Image , the value of this property is given by the MediaURI element that is a descendant of that element. For assets without an appropriate RelatedMaterial element, the value of this property shall be undefined .
readonly Integer length
The number of items in the current page. If getPage() has not yet been called, the value of this property shall be undefined .
readonly Integer currentPage
The page number of the currently-available results, as specified in the last call to getPage() . If getPage() has not yet been called, the value of this property shall be undefined .

readonly Integer pageSize
The number of items that were requested from the content catalogue in a call to <code>getPage()</code> . This may be different from the number of items that are available (e.g. the last page in the collection). If <code>getPage()</code> has not yet been called, the value of this property shall be undefined .

readonly Integer totalSize
The total number of items in the folder. This may be undefined until <code>getPage()</code> has been called. The value of this property may be given by the <code>numOfItems</code> attribute of the <code>GroupInformation</code> element representing this folder.

7.5.6.3 Methods

Object item(Integer index)		
Description	Return the item at position index in the current page, or undefined if no item is present at that position. This function shall only return objects that are instances of <code>CODAsset</code> , <code>CODFolder</code> , or <code>CODService</code> . Applications shall be able to access items in the collection using array notation instead of calling this method directly.	
Arguments	index	The index into the collection.

void getPage(Integer page, Integer pageSize)		
Description	Retrieve one page of the folder's contents. The application shall be notified by an event targeted at the folder's parent content catalogue when the data is available. Calls to this method shall cancel any outstanding requests.	
Arguments	page	The number of the page for which data should be retrieved, indexed from zero.
	pageSize	The size of the page.

void abort()	
Description	Abort the current request for a new page of folder contents. Any results for this folder shall be removed (i.e. the value of the <code>length</code> property will be 0 and any calls to the <code>item()</code> method shall return undefined).

7.5.7 The CODAsset class

7.5.7.1 General

The `CODAsset` represents a piece of CoD content that can be purchased and played. A `CODAsset` object may refer to a bundle of content items that are purchased together, but which can only be played individually.

Some fields of a `CODAsset` object may not be populated until an application requests them; in this case the data may be fetched asynchronously from a server. Fields where the data has not been fetched from the server shall have a value of **undefined**. Fields for which data is not available on the server shall have a value of **null**.

NOTE The `lookupMetadata()` method has been removed from this class.

7.5.7.2 Properties

readonly Integer type
The type of the item, used to distinguish between the types of objects that may be contained in a folder in a CoD catalogue. This property shall always have the value 1 for CoD assets.
readonly String uri
The CRID of the asset. The value of this property is given by the programId attribute of the BCG ProgramInformation element that describes the asset.
readonly String name
The title of the asset that will be displayed to the user. The value of this property is given by the BCG Title element that is a child of the asset's BasicDescription element.
readonly String description
A description of the asset, for display to an end user. The value of this property is given by the BCG Synopsis element that is a child of the asset's BasicDescription element.
readonly StringCollection genres
A collection of genres that describe this asset. Genres are represented by the values of any Name elements that are children of Genre elements in the asset's description.
readonly ParentalRatingCollection parentalRatings
The parental rating value of the asset. This information will be read from the ParentalGuidance element of an asset's description, if present.
readonly Boolean blocked
Flag indicating whether the asset is blocked due to parental control settings (i.e. whether its parental rating value exceeds the current system threshold).
readonly Boolean locked
Flag indicating whether the current state of the parental control system prevents the asset from being viewed (e.g. a correct parental control PIN has not been entered to allow the item to be viewed).
readonly String thumbnailUri
The URI of an image associated with this asset. For assets whose BCG description contains a RelatedMaterial element indicating a relationship of Promotional Still Image, the value of this property is given by the MediaURI element that is a descendant of that element. For assets without an appropriate RelatedMaterial element, the value of this property shall be undefined.

readonly string price	
The price of the asset, in a form that can be displayed to the user. The value of this property is the concatenation of the value of the Price element that is a child of a PurchaseItem element in the asset's description and the value of the Price element's Currency attribute. For example, a Price element of <pre><Price currency="JPY">500</Price></pre> would give the value 500 JPY for this field. Implementations may replace the currency code with the appropriate currency symbol (e.g. ¥).	

readonly Integer rentalPeriod	
The period for which the asset can be rented, in hours. For assets descriptions containing a Purchase element with a PurchaseType of urn:tva:metadata:cs:PurchaseTypeCS:2004:playForPeriod, the value of this property is derived from the QuantityUnit and QuantityRange elements that are children of that Purchase element. If a Purchase element with the appropriate PurchaseType is not present, the value of this field shall be undefined.	

readonly Integer playCount	
The number of plays allowed for this asset when it is purchased. For assets descriptions containing a Purchase element with a PurchaseType of urn:tva:metadata:cs:PurchaseTypeCS:2004:playCounts, the value of this property is derived from the QuantityUnit and QuantityRange elements that are children of that Purchase element. If a Purchase element with the appropriate PurchaseType is not present, the value of this field shall be undefined.	

readonly Integer duration	
The duration of the asset, in seconds. The value of this property is given by the BCG Duration element that is a child of the asset's BasicDescription element.	

readonly string previewUri	
The URI used to refer to a preview of the asset. For assets whose BCG description contains a RelatedMaterial element indicating a relationship of Trailer or Preview, the value of this property is given by the MediaURI element of the MediaLocator contained in that element. For assets without an appropriate RelatedMaterial element, the value of this property shall be undefined.	

readonly BookmarkCollection bookmarks	
A collection of the bookmarks set in a recording. If no bookmarks are set, the collection shall be empty.	

7.5.7.3 Methods

Boolean isReady()	
Description	Check whether sufficient information is available to make a purchase or play the asset. Due to the asynchronous nature of CoD catalogues, not all of the information required to play or purchase a CoD asset may have been received by the OITF at any given time. If all of the required information is available, this method shall return true. Otherwise, this method shall request the missing information and return false. When the information is available, the application shall be notified via a ContentAction event with the reason code 1.

7.5.8 The CODService class

7.5.8.1 General

The `CODService` class is a subclass of `CODFolder` that represents a subscription CoD service. A subscription CoD service is similar to a folder, except that:

- The service shall be purchased in its entirety, rather than purchasing individual items from the service.
- Business rules may prevent browsing of the content within a service unless the service has already been purchased.

A `CODService` may contain a number of assets, folders and services.

NOTE The `lookupMetadata()` method and `uid` property has been removed from this class.

7.5.8.2 Properties

readonly Integer length
The number of items in the current page of the service.
readonly Integer currentPage
The page number of the currently-available results, as specified in the last call to <code>getPage()</code> . If <code>getPage()</code> has not yet been called, the value of this property shall be undefined .
readonly Integer pageSize
The number of items that were requested from the content catalogue in a call to <code>getPage()</code> . This may be different from the number of items that are available (e.g. the last page in the collection). If <code>getPage()</code> has not yet been called, the value of this property shall be undefined .
readonly Integer totalSize
The total number of items in the service. This may be undefined until <code>getPage()</code> has been called. The value of this property may be given by the <code>numOfItems</code> attribute of the <code>GroupInformation</code> element representing this folder.
readonly Integer type
The type of the item, used to distinguish between the types of objects that may be contained in a folder in a CoD catalogue. This property shall always have the value 2 for a CoD service.
readonly String uri
The URI used to refer to the service. The value of this property is given by the BCG <code>ServiceURL</code> element that is a child of the <code>ServiceInformation</code> element that describes the service.
readonly String name
The name of the service that will be displayed to the user. The value of this property is given by the BCG <code>Name</code> element that is a child of the <code>ServiceInformation</code> element describing the service.
readonly String description
A description of the service, for display to an end user. The value of this property is given by the BCG <code>ServiceDescription</code> element that is a child of the <code>ServiceInformation</code> element describing the service.

readonly string thumbnailUri
The URI of an image associated with this service. The value of this property is derived from the value of the first Logo element that is a child of the BCG ServiceInformation element describing the service. If this element specifies anything other than the URL of an image, the value of this property shall be undefined. Alternatively, for services whose BCG description contains a RelatedMaterial element indicating a relationship of Promotional Still Image, the value of this property is given by the MediaURI element of the MediaLocator contained in that element. For assets without an appropriate RelatedMaterial or Logo element, the value of this property shall be undefined.

readonly string previewUri
The URI used to refer to a preview of the content. For services whose BCG description contains a RelatedMaterial element indicating a relationship of Trailer or Preview, the value of this property is given by the MediaURI element of the MediaLocator contained in that element. For services without an appropriate RelatedMaterial element, the value of this property shall be undefined.

7.5.8.3 Methods

Boolean isReady()	
Description	Check whether sufficient information is available to make a purchase. Due to the asynchronous nature of CoD catalogues, not all of the information required to play or purchase a CoD service may have been received by the OITF at any given time. If all of the required information is available, this method shall return true. Otherwise, this method shall request the missing information and return false. When the information is available, the application shall be notified via a ContentAction event with the action code 1.

Object item(Integer index)			
Description	Return the item at position index in the current page, or undefined if no item is present at that position. This function shall only return objects that are instances of CODAsset, CODFolder, or CODService. Applications shall be able to access items in the collection using array notation instead of calling this method directly.		
Arguments	<table> <tr> <td>index</td><td>The index into the collection.</td></tr> </table>	index	The index into the collection.
index	The index into the collection.		

void getPage(Integer page, Integer pageSize)					
Description	Retrieve one page of the services contents. The application shall be notified by an event targeted at the services parent content catalogue when the data is available. Calls to this method shall cancel any outstanding requests.				
Arguments	<table> <tr> <td>page</td><td>The number of the page for which data should be retrieved, indexed from zero.</td></tr> <tr> <td>pageSize</td><td>The size of the page.</td></tr> </table>	page	The number of the page for which data should be retrieved, indexed from zero.	pageSize	The size of the page.
page	The number of the page for which data should be retrieved, indexed from zero.				
pageSize	The size of the page.				

void abort()	
Description	Abort the current request for a new page of contents. Any results shall be removed (i.e. the value of the length property will be 0 and any calls to the item() method shall return undefined).

7.6 Content service protection API

7.6.1 General

The requirements in 7.6 shall apply to OITF and/or server devices which have indicated support for DRM protection by providing one or more <drm> elements as specified in 9.3.11.

7.6.2 The application/oipfDrmAgent embedded object

7.6.2.1 General

An OITF shall support a non-visual embedded object of type "application/oipfDrmAgent", with the following JavaScript API, to enable in-session message exchange from the web page with an underlying DRM agent.

Access to the functionality of the application/oipfDrmAgent embedded object shall adhere to the security requirements as defined in 10.1.

NOTE Annex B provides a clarification to the browser interaction model when dealing with protected content.

7.6.2.2 Properties

function onDRMMessageResult (String msgID, String resultMsg, Integer resultCode)																										
The function that is called when the underlying DRM agent has a result message to report to the current HTML document as a consequence of a call to sendDRMMessage . The specified function is called with three arguments msgID , resultMsg and resultCode which are defined as follows:																										
- String msgID – identifies the original message which has led to this resulting message. - String resultMsg – DRM system specific result message. - Integer resultCode – result code. Valid values include: <table border="1"> <thead> <tr> <th>Result message</th><th>Description</th><th>Semantics</th></tr> </thead> <tbody> <tr> <td>0</td><td>Successful</td><td>The action(s) requested by sendDRMMessage() completed successfully.</td></tr> <tr> <td>1</td><td>Unknown error</td><td>sendDRMMessage() failed because an unspecified error occurred.</td></tr> <tr> <td>2</td><td>Cannot process request</td><td>sendDRMMessage() failed because the DRM agent was unable to complete the request.</td></tr> <tr> <td>3</td><td>Unknown MIME type</td><td>sendDRMMessage() failed, because the specified MIME Type is unknown for the specified DRM system indicated in the DRMSYSTEMID.</td></tr> <tr> <td>4</td><td>User consent needed</td><td>sendDRMMessage() failed because user consent is needed for that action.</td></tr> <tr> <td>5</td><td>Unknown DRM system</td><td>sendDRMMessage() failed, because the specified DRM System in DRMSYSTEMID is unknown.</td></tr> <tr> <td>6</td><td>Wrong format</td><td>sendDRMMessage() failed, because the message in msg has the wrong format.</td></tr> </tbody> </table>			Result message	Description	Semantics	0	Successful	The action(s) requested by sendDRMMessage() completed successfully.	1	Unknown error	sendDRMMessage() failed because an unspecified error occurred.	2	Cannot process request	sendDRMMessage() failed because the DRM agent was unable to complete the request.	3	Unknown MIME type	sendDRMMessage() failed, because the specified MIME Type is unknown for the specified DRM system indicated in the DRMSYSTEMID .	4	User consent needed	sendDRMMessage() failed because user consent is needed for that action.	5	Unknown DRM system	sendDRMMessage() failed, because the specified DRM System in DRMSYSTEMID is unknown.	6	Wrong format	sendDRMMessage() failed, because the message in msg has the wrong format.
Result message	Description	Semantics																								
0	Successful	The action(s) requested by sendDRMMessage() completed successfully.																								
1	Unknown error	sendDRMMessage() failed because an unspecified error occurred.																								
2	Cannot process request	sendDRMMessage() failed because the DRM agent was unable to complete the request.																								
3	Unknown MIME type	sendDRMMessage() failed, because the specified MIME Type is unknown for the specified DRM system indicated in the DRMSYSTEMID .																								
4	User consent needed	sendDRMMessage() failed because user consent is needed for that action.																								
5	Unknown DRM system	sendDRMMessage() failed, because the specified DRM System in DRMSYSTEMID is unknown.																								
6	Wrong format	sendDRMMessage() failed, because the message in msg has the wrong format.																								

function onDRMSystemStatusChange (String DRMSYSTEMID)
The function that is called when the status of a DRM system changes.
The specified function is called with one argument DRMSYSTEMID which is defined as follows:
String DRMSYSTEMID – argument that specifies the DRM System ID of the DRM system that generated the event as defined by element DRMSYSTEMID in Table 9 of IEC 62766-3:2016.

function onDRMSystemMessage(String msg, String DRMSystemID)
The function that is called when the underlying DRM system has a message to report to the current HTML document. The specified function is called with two arguments msg and DRMSystemID and msg which are defined as follows: • String msg – DRM system specific message • String DRMSystemID – argument that specifies the DRM System ID of the DRM system that generated the event as defined by element DRMSystemID in Table 9 of IEC 62766-3:2016.

7.6.2.3 Methods

String sendDRMMessage (String msgType, String msg, String DRMSysID)		
Description	Send message to the DRM agent, using a message type as defined by the DRM system. Returns a unique ID to identify the message, to be passed as the 'msgID' argument for the callback function registered through onDRMMessageResult. This is an asynchronous method. Applications will be notified of the results of the operation via events dispatched to onDRMMessageResult and corresponding DOM events.	
Arguments	msgType	A globally unique message type as defined by the DRM system, for example: application/vnd.marlin.drm.actiontoken+xml (i.e. MIME type of Marlin Action Token). Valid values for the msgType parameter include the MIME types described in Annex C of IEC 62766-7:2017.
	msg	The message to be provided to the underlying DRM agent formatted according to the message type as indicated by attribute msgType. Valid format for the msg parameter are message formats described in Annex C of IEC 62766-7:2017.
	DRMSysID	DRMSysID as defined by element DRMSysID in Table 9 of IEC 62766-7. For example, for Marlin, the DRMSysID value is "urn:dvb:casystemid:19188". In the case that parameter msgType indicates a CSPG-CI+ message as described in 4.3.4.5.1.1.2 of IEC 62766-7:2017 or an embedded CSPG message (see Annex F of IEC 62766-7:2017), the DRMSysID parameter shall be specified. Otherwise, the value may be null.

Integer DRMSystemStatus (String DRMSystemID)		
Description	Returns the status of the indicated DRM system, as defined below:	
	Value	Description
	0	READY
	1	UNKNOWN
	2	INITIALISING
	3	ERROR
		Semantics
		The DRM system is fully initialised and ready.
		Unknown DRM system.
		The DRM system is initialising and not ready to start communicating with the application.
		There is a problem with the DRM system. It may be possible to communicate with it to obtain more information.
Arguments	DRMSystemID	The DRM System ID of the DRM system that is being queried as defined by the element DRMSystemID in Table 9 of IEC 62766-3:2016. For example, for Marlin, the DRMSystemID value is "urn:dvb:casystemid:19188".

Boolean canPlayContent(String DRMPPrivateData, String DRMSystemID)		
Description	Checks the local availability of a valid license for playing a protected content item. The function returns true if there is a valid license available locally that may allow playing the content. For example the actual playing may be blocked due to other constraints (e.g. video/audio output restrictions on selected output). The DRMPPrivateData may be retrieved by the application via a means out of scope of this standard (e.g. retrieved from Service Platform, or from a manifest file). For already downloaded content, the private data may be retrieved via the <code>getDRMPPrivateData()</code> method of the <code>Download</code> class. In case the download is triggered through a content access download descriptor, the private data may be retrieved from the <code>drmControl</code> property.	
Arguments	DRMPPrivateData	DRM proprietary private data.
	DRMSystemID	DRMSystemID as defined by element DRMSystemID in Table 9 of IEC 62766-3:2016. For example, for Marlin, the DRMSystemID value is "urn:dvb:casystemid:19188".

Boolean canRecordContent(String DRMPPrivateData, String DRMSystemID)		
Description	Checks the local availability of a valid license for recording a protected content item. The function returns true if there is a valid license available locally that may allow recording the content. The DRMPPrivateData may be retrieved by the application via a means out of scope of this standard (e.g. retrieved from Service Platform, or from a manifest file).	
Arguments	DRMPPrivateData	DRM proprietary private data.
	DRMSystemID	DRMSystemID as defined by element DRMSystemID in Table 9 of IEC 62766-3. For example, for Marlin, the DRMSystemID value is "urn:dvb:casystemid:19188".

7.6.2.4 Events

For the intrinsic events listed in the table below, a corresponding DOM event shall be generated in the following manner:

Intrinsic event	Corresponding DOM event	DOM Event properties
onDRMMessageResult	DRMMessageResult	Bubbles: No Cancellable: No Context Info: msgID, resultMsg, resultCode
onDRMSystemStatusChange	DRMSystemStatusChange	Bubbles: No Cancellable: No Context Info: DRMSystemID
onDRMSystemMessage	DRMSystemMessage	Bubbles: No Cancellable: No Context Info: msg, DRMSystemID

NOTE The above DOM events are directly dispatched to the event target, and will not bubble nor capture. Applications should NOT rely on receiving these events during the bubbling or the capturing phase. The `addEventListener()` method should be called on the `application/oipfDrmAgent` object itself. The third parameter of `addEventListener`, i.e. "useCapture", will be ignored.

7.7 Gateway discovery and control APIs

7.7.1 General

The `application/oipfGatewayInfo` object shall provide the information of the gateway and subsequently interact with the gateway (e.g. IMS Gateway, Application Gateway, CSPG-

CI+ Gateway and CSPG-DTCP Gateway) as defined in 4.3. The OITF shall support the gateway discovery and control through the use of the following non-visual embedded object:

```
<object id="gatewayinfo" type="application/oipfGatewayInfo">
```

Access to the functionality of the application/oipfGatewayInfo embedded object is privileged and shall adhere to the security requirements defined in 10.1.

7.7.2 The application/oipfGatewayInfo embedded object

7.7.2.1 Properties

readonly Boolean isIGDiscovered
Readonly property that indicates whether an IMS Gateway is discovered or not. NOTE This property was formerly referred to as IGDiscovery .
readonly Boolean isAGDiscovered
Readonly property that indicates whether an Application Gateway is discovered or not. NOTE This property was formerly referred to as AGDiscovery .
readonly Boolean isCSPGCIPlusDiscovered
Readonly property that indicates whether a CSPG-CI+ Gateway is discovered or not. NOTE This property was formerly referred to as cspGatewayDiscovery . The former cspGatewayDiscovery property is now replaced with isCSPGCIPlusDiscovered for CSPG-CI+ case and isCSPGDTCPCovered for CSPG-DTCP case.
readonly Boolean isCSPGDTCPCovered
Readonly property that indicates whether a CSPG-DTCP Gateway is discovered or not. NOTE This property was formerly referred to as cspGatewayDiscovery . The former cspGatewayDiscovery property is now replaced with isCSPGCIPlusDiscovered for CSPG-CI+ case and isCSPGDTCPCovered for CSPG-DTCP case.
readonly String igURL
Readonly property that indicates the base Gateway's URL for interacting between an OITF and an IMS Gateway.
readonly String agURL
Readonly property that indicates the base Gateway's URL for interacting between an OITF and an Application Gateway.
readonly String cspgDTCPUURL
Readonly property that indicates the base Gateway's URL for interacting between an OITF and an CSPG-DTCP Gateway. NOTE This property was formerly referred to as cspGatewayURL , which was relevant for CSPG-DTCP case only.
Integer interval
Read-write property that specifies the periodic interval time(seconds) to discover the gateways. When the interval property is set, a UPnP Discovery mechanism is executed.

function onDiscoverIG()

The function that shall be called when an IMS Gateway is discovered or lost by the OITF which uses a UPnP Discovery mechanism described in IEC 62766-4-1:2017, 11.2. The actual status of the gateway (discovered or not) can be determined by reading the `isIGDiscovered` property.

The specified function is called with no arguments.

function onDiscoverAG()

The function that shall be called when an Application Gateway is discovered or lost by the OITF which uses a UPnP Discovery mechanism described in IEC 62766-4-1:2017, 11.3. The actual status of the gateway (discovered or not) can be determined by reading the `isAGDiscovered` property.

The specified function is called with no arguments.

function onDiscoverCSPGDTCP()

The function that shall be called when an CSPG-DTCP Gateway is discovered or lost by the OITF. The CSPG-DTCP gateway shall be discovered using a UPnP Discovery mechanism described in IEC 62766-4-1:2017, 11.4. The actual status of the gateway (discovered or not) can be determined by reading the `isCSPGDTCPDiscovered` property.

The specified function is called with no arguments.

NOTE This property was formerly referred to as **onDiscoverCSPG**. The former **onDiscoverCSPG** property is now replaced with **onDiscoverCSPGCIPlus** for CSPG-CI+ case and **onDiscoverCSPGDTCP** for CSPG-DTCP case.

readonly Boolean isIGSupported

Readonly property that indicates whether an IMS Gateway is supported or not.

readonly Boolean isAGSupported

Readonly property that indicates whether an Application Gateway is supported or not.

readonly Boolean isCSPGCIPlusSupported

Readonly property that indicates whether a CSPG-CI+ Gateway is supported or not.

readonly Boolean isCSPGDTCPSupported

Readonly property that indicates whether a CSPG-DTCP Gateway is supported or not.

function onDiscoverCSPGCIPlus()

The function that shall be called when a CSPG-CI+ Gateway is discovered or lost by the OITF (including any change to the DRM systems supported by that gateway). The CSPG-CI+ Gateway shall be discovered as defined in IEC 62766-7. The actual status of the gateway (discovered or not) can be determined by reading the `isCSPGCIPlusDiscovered` property.

The specified function is called with no arguments.

NOTE This property was formerly referred to as **onDiscoverCSPG**. The former **onDiscoverCSPG** property is now replaced with **onDiscoverCSPGCIPlus** for CSPG-CI+ case and **onDiscoverCSPGDTCP** for CSPG-DTCP case.

readonly StringCollection CSPGCIPlusDRMType

Readonly property that indicates the list of CA Systems supported by the CSPG-CI+ Gateway under the form of URN with the DVB CASystemID (16-bit number) in there. Each element of **CSPGCIPlusDRMType** shall be signalled by prefixing the decimal number format of CA_System_ID with "urn:dvb:casystemid:".

7.7.2.2 Methods

Boolean isIGSupportedMethod (String methodName)		
Description	Shall return true when the IG supports the method specified in the 'methodName' argument. If the function returns false , it indicates that IG does not support the specified method.	
Arguments	methodName	The name of the method to be checked for support.

7.7.2.3 Events

For the intrinsic events listed in the table below, a corresponding DOM event shall be generated, in the following manner:

Intrinsic event	Corresponding DOM event	DOM Event properties
onDiscoverIG	DiscoverIG	Bubbles: No Cancellable: No
onDiscoverAG	DiscoverAG	Bubbles: No Cancellable: No
onDiscoverCSPGDTCP	DiscoverCSPGDTCP	Bubbles: No Cancellable: No
onDiscoverCSPGCIPlus	DiscoverCSPGCIPlus	Bubbles: No Cancellable: No

NOTE The above DOM events are directly dispatched to the event target, and will not bubble nor capture. Applications should not rely on receiving these events during the bubbling or the capturing phase. Applications that use DOM event handlers shall call the **addEventListener()** method on the **application/oipfGatewayInfo** object. The third parameter of **addEventListener**, i.e. "useCapture", will be ignored.

7.8 Communication services APIs

7.8.1 General

If an OITF has indicated support for the control of its Communication Services functionality by a server by stating **<communicationServices>true</communicationServices>** as defined in 9.3.10 in its capability description, the OITF shall support communication services through the use of the following non-visual object:

```
<object type="application/oipfCommunicationServices"/>
```

The Communication Services API provides the necessary JavaScript methods to register new users. It also provides methods to register users (**registerUser**), along with the supported feature tags. A method **getRegisteredUsers** is also defined to view all the registered users. A method **getAllUsers** retrieves all users provisioned in the IG. Once registered it is possible to switch users for using communication services by using method **setUser**.

A property is defined to view the current user to be used for a service (**currentUser**).

In order to handle the out-of-session communication services notifications, namely, the new dialogues, there is a method for subscribing to these events (**subscribeNotification**). All new dialogues are communicated through a callback function (**onNotification**) to the application instance performing the subscription.

The Communication Services APIs apply only to privileged applications and shall adhere to the security model as defined in Clause 10.

7.8.2 The application/oipfCommunicationServices embedded object

7.8.2.1 Constants

Name	Value	Use
STATE_REGISTERED	0	Specifies that the user has been successfully registered (not subscribed to registration event). This also represents the state when the registration event subscription has been terminated for some reason by network.
STATE_REGISTERED_SUBSCRIPTION_PENDING	1	Indicates that user is registered successfully but the subscription-state for the registration event indicates a status of "pending".
STATE_REGISTERED_SUBSCRIPTION_ACTIVE	2	Specifies that the user has been successfully registered and subscribed to registration event (i.e. subscription-state for registration event indicates a status of "active").
STATE_DEREGISTERED	3	Specifies that the user has been successfully deregistered. This can be result of network initiated/locally initiated deregistration request.
STATE_FAILURE	4	Represents a failure condition.

7.8.2.2 Properties

<pre>function onNotification(String responseHeaders, String msgText, Document msgXML)</pre>
This function is called on the application which called subscribeNotification when an unsolicited notification arrives. The application will be notified of all notifications corresponding to any of the subscribed-to feature tags regardless of which application subscribed to it. The specified function is called with 3 arguments. • String responseHeaders – The concatenated list of all HTTP headers, as a single string, with each header line separated by a U+000D (CR) U+000A (LF) pair excluding the status line. In absence of HNI-IGI interface, the responseHeaders will be a concatenated list all SIP headers, as a single string, with each header line separated by a U+000D (CR) U+000A (LF) pair excluding the status line. • String msgText – the response entity body as a string, as defined in the XMLHttpRequest specification as referenced in IEC 62766-5-2. • Document msgXML – the response entity body as a Document, as defined in the XMLHttpRequest specification as referenced in IEC 62766-5-2.

function onNotificationResult(Integer resultMsg)		
This function is called with return result from the subscribeNotification method.		
This function is not invoked in the case when there is no re-registration as part of subscribeNotification .		
The specified function is called with a single argument – resultMsg .		
Integer resultMsg – result message from performing subscribeNotification method.		
Result message	Description	Semantics
0	Successful	The action performed by the underlying functionality was successful.
1	Unknown error	The action performed by the underlying functionality failed because an unspecified error occurred.
2	Wrong user credentials	The user credentials was not accepted by the server.
3	The user doesn't exist.	The user id doesn't exist in the local user table.

function onRegistrationContextUpdate(String user, Integer state, Integer errorCode)		
This function is called with return result from the methods registerUser and deRegisterUser . In addition, the function is also called whenever there is an update to the registration status of specified user.		
The specified function is called with 3 arguments – user , state and errorCode .		
- String user – The IMPU of the user. - Integer state – The current state of the registration as indicated using the constant values defined in 7.8.2.1. - Integer errorCode – In case of STATE_FAILED state, provides more information on reason for failure.		
errorCode	Description	Semantics
1	Unknown error	The action performed by the underlying functionality failed because an unspecified error occurred.
2	Wrong user credentials	The user credentials were not accepted by the server. The DAE may request from the user a new PIN which can then be used to perform a new registerUser with the provided PIN.
3	The user doesn't exist.	The user id doesn't exist in the local user table.

readonly UserData currentUser
The currentUser property represents the public user identity which is being used or shall be used for HNI-IGI communication. If not set, then the default user shall be used or indicated. It shall be set to the default user if a user has not been explicitly set using the setUser() method.

7.8.2.3 Methods

UserDataCollection getRegisteredUsers()	
Description	Return all the users that are currently registered with the IG.

Void registerUser(String userId, String pin)		
Description	This method performs user registration to the network. Results from this method is sent to the callback method onRegistrationContextUpdate.	
Arguments	userId	The user identifier represents the public user identity or IMPU.
	pin	The pin is optional and carries the password to be used towards the IG in case of HTTP Digest over HNI-IGI interface or SIP Digest if there is no HNI-IGI. If pin is not specified, then the default user password shall be used. The pin used for digest authentication is limited to the HNI-IGI interface with the IG and shall not impact the HTTP Digest requests from within the DAE application. Support for this parameter is not applicable for non-native HNI-IGI.

void deRegisterUser(String userId)		
Description	The indicated user is de-registered. Any sessions that may be open are closed. De-registration of default user has no effect nor de-registration of any users registered from a native application in the OITF. Results from this method is sent to the callback method onRegistrationContextUpdate.	
Arguments	userId	The user identifier represents the public user identity or IMPU.

UserDataCollection getAllUsers()	
Description	Return all the users that are currently provisioned in the IG. The first entry in the collection is the default user. The users are retrieved according to IEC 62766-4-1:2017, 6.4.6.4.

Boolean setUser(String userId)		
Description	When invoked, any ongoing sessions for the current user shall be closed. If setUser is unsuccessful due to user not being registered, it is necessary to first register the user and wait for a successful response to the onRegistrationContextUpdate callback function. If the user gets deregistered (either by the local application or by the network), any ongoing sessions for the user shall be closed. The default user shall be automatically assumed for all services until overridden again by the setUser method.	
Argument	userId	The user identifier represents the public user identity or IMPU.

void subscribeNotification(FeatureTagCollection featureTagCollection, Boolean performUserRegistration)		
Description	This method subscribes for new IMS out-of-session dialogues for the indicated application for the currently active user. The notification shall be notified using onNotification callback method. If the application that made the subscription closes then there is an automatic un-subscription to new notifications. Otherwise it is possible to perform unsubscribeNotification. Any new dialogues shall be notified over the callback method onNotification.	
Arguments	featureTagCollection	The featureTagCollection object of the DAE application. If the value of this argument is NULL then all dialogs shall be reported.
	performUserRegistration	If this is true a new user registration is required. should be set to false if it is know that other applications will be registered shortly This parameter is ignored in the case when the filtering of notifications is done locally. In this case, the initial registration for active user will include all feature tags.

void unsubscribeNotification()	
Description	The DAE application calling this method will be de-registered for notifications. Associated feature tag(s) for the DAE application are removed from the featureTagCollection object for the user. A re-registration will be performed for the corresponding user if notifications are not locally filtered. Results from this method is sent to the callback method <code>onNotificationResult</code>.

7.8.2.4 Events

For the intrinsic events listed in the table below, a corresponding DOM event shall be generated, in the following manner:

Intrinsic event	Corresponding DOM event	DOM Event properties
<code>onNotificationResult</code>	<code>NotificationResult</code>	Bubbles: No Cancellable: No Context Info: <code>resultMsg</code>
<code>onNotification</code>	<code>Notification</code>	Bubbles: No Cancellable: No Context Info: <code>callId, contact, from, to</code>
<code>onRegistrationContextUpdate</code>	<code>RegistrationContextUpdate</code>	Bubbles: No Cancellable: No Context Info: <code>user, state, errorCode</code>

These DOM events are directly dispatched to the event target, and will not bubble nor capture. Applications should not rely on receiving these events during the bubbling or the capturing phase. Applications that use DOM event handlers shall call the `addEventListener()` method on the `application/oipfCommunicationServices` object itself. The third parameter of `addEventListener`, i.e. "useCapture", will be ignored.

7.8.3 Extensions to application/oipfCommunicationServices for presence and messaging services

7.8.3.1 General

If a client has indicated support for the control of its presence and messaging functionality by a server by stating `<presenceMessaging>true</presenceMessaging>` as defined in 9.3.10 in its capability description, the client shall support Communication Services through the use of the following non-visual embedded object:

```
<object type="application/oipfCommunicationServices"/>
```

The presence and messaging API provides for instant messaging, presence and contact list services. The messages can either be in a chat session using MSRP (see IEC 62766-4-1) or page mode messages sent without a session. The support of presence and messaging shall follow the OMA specifications OMA-TS-Presence_SIMPLE_XDM-V1_1-20080627-A and Open Mobile Alliance OMA-TS-SIMPLE_IM-V1_0-20080820-D.

The Communication Services API shall be supported in combined OITF and IG deployment cases. It may be supported in other deployment cases. The use of the HNI-IGI interface is optional between the OITF and IG when these are co-deployed.

7.8.3.2 Properties

function onIncomingMessage(String fromURI, String msg, Integer cid)
The function that is called when an incoming chat message is received for the active user.
The specified function is called with 3 arguments:
- String fromURI – The sender address of the message. - String msg – The text message sent by the remote peer. - Integer cid – Chat session identifier, either the same as one received from the openSession() method or new if session is started by remote peer. Empty identifier if message is sent without a session.

function onContactStatusChange(String remoteURI, Integer state)
This function is called when status has changed for a contact in the contact list or a user used with the method subscribeToStatus().
The specified function is called with two arguments:
- String remoteURI – The user address for which the status has changed. - Integer state – Set to 1 if the user is present, and 0 if not. Other values may be defined in the future.

function onNewWatcher(String remoteURI)
This function is called when a remote URI is requesting watcher authorization of the local user's presentity.
The specified function is called with one argument:
String remoteURI – The remote user address which requested watcher authorization.

7.8.3.3 Methods

Integer openChatSession(String toURI)		
Description	Opens a chat session with a remote user. Returns an integer identifier for the chat session to be used when a message is sent in the chat session or to match when incoming message is received.	
Arguments	toURI	The address of the remote chat user.

void sendMessageInSession (Integer cid, String msg)		
Description	Sends a new text message in a chat session. The chat can either be started by the user by calling the method <code>openChatSession()</code> or can be a session received in the <code>onIncomingMessage</code> callback function.	
Arguments	cid	The chat session identifier.
	msg	Text message to send.

void closeChatSession(Integer cid)		
Description	Closes a chat session.	
Arguments	cid	The chat session identifier.

void sendMessage(String toURI, String msg)		
Description	Sends a new text message to a remote peer without starting a session.	
Arguments	toURI	The address of the remote chat user.
	msg	Text message to send.

void setStatus(Integer state)		
Description	Sets the presence state of the local user.	
Arguments	state	Set to 1 if the user is present, and 0 if not. Other values may be defined in the future.

void subscribeToStatus(String remoteURI)		
Description	Subscribe to status for a remote user.	
Arguments	remoteURI	The address of the remote user.

ContactCollection getContacts()	
Description	Get the users contact list.

void allowContact(String remoteURI)		
Description	Allows the watcher authorization to subscribe to the local user's presentity.	
Arguments	remoteURI	The address of the remote user.

void blockContact(String remoteURI)		
Description	Blocks the watcher authorization to subscribe to the local user's presentity.	
Arguments	remoteURI	The address of the remote user.

Boolean createContactList(String contactListUri, ContactCollection contacts)		
Description	Creates a contact list.	
Arguments	contactListUri	The public user identity or IMPU of the contact list.
	contacts	The collection of contact objects representing the members of the list.

ContactCollection getContacts(String contactListUri)		
Description	Get the users in the specified contact list.	
Arguments	contactListUri	The public user identity or IMPU of the contact list.

Boolean addToContactList(String contactListUri, Contact member)		
Description	Updates the specified contact list by adding a new member to that list.	
Arguments	contactListUri	The public user identity or IMPU of the contact list to be updated.
	member	The new contact to be added to the list.

Boolean removeFromContactList(String contactListUri, Contact member)		
Description	Updates the specified contact list by removing specified member from that list.	
Arguments	contactListUri	The public user identity or IMPU of the contact list to be updated.
	member	The new contact to be removed from the list

Boolean deleteContactList(String contactListUri)		
Description	Deletes the specified contact list.	
Arguments	contactListUri	The public user identity or IMPU of the contact list to be deleted

void allowAllContacts(String domain)		
Description	Allows all watchers belonging to specified domain authorization to subscribe to local user's presentity. If null , then all contacts will be allowed.	
Arguments	domain	Watchers belonging to this domain are authorized to subscribe. If null , then all watchers are authorized to subscribe irrespective of domain.

void blockAllContacts(String domain)		
Description	Blocks all watchers belonging to specified domain from subscribing to local user's presentity. If null , then all contacts will be blocked.	
Arguments	domain	Watchers belonging to this domain are denied authorization to subscribe. If null , then all watchers are blocked from subscribing irrespective of domain.

7.8.3.4 Events

For the intrinsic events listed in the table below, a corresponding DOM event shall be generated in the following manner:

Intrinsic event	Corresponding DOM event	DOM Event properties
onIncomingMessage	IncomingMessage	Bubbles: No Cancellable: No Context Info: fromURI, msg, cid
onContactStatusChange	ContactStatusChange	Bubbles: No Cancellable: No Context Info: remoteURI, present
onNewWatcher	NewWatcher	Bubbles: No Cancellable: No Context Info: remoteURI

These DOM events are directly dispatched to the event target, and will not bubble nor capture. Applications should not rely on receiving these events during the bubbling or the capturing phase. Applications that use DOM event handlers shall call the `addEventListener()` method on the `application/oipfCommunicationsServices` object itself. The third parameter of `addEventListener`, i.e. "useCapture", will be ignored.

7.8.4 The UserData class – Properties

readonly String userId
The user identifier represents the public user identity or IMPU.

readonly FeatureTagCollection featureTags
The feature tag data is optional. It carries a collection of feature tag objects associated with an application. For example, the feature tag may be an ICSI or IARI or a feature tag identifying the service for. an incoming instant messages. The object includes feature tags related to both DAE and native applications in OITF.

readonly string friendlyName
The friendly name for the user. Used as display name in outgoing messages.

7.8.5 The UserDataCollection class

```
typedef Collection<UserData> UserDataCollection
```

The UserDataCollection class represents a collection of UserData objects. See Annex I for the definition of the collection template.

7.8.6 The FeatureTag class – Properties

readonly String featureTag
A string containing a featureTag value associated to an application. The featureTag value may have a value of null when used with the subscribeNotification() method on the application/oipfCommunicationServices object. This indicates that all dialogues are reported. The feature tag shall populate the X-OITF- headers as specified in ETSI TS 183 063 5.6.2, Open Mobile Alliance OMA-TS-SIMPLE_IM-V1_0-20080820-D, 3GPP TS 24.229, IETF RFC 3840 and IETF RFC 3841.

7.8.7 The FeatureTagCollection class

```
typedef Collection<FeatureTag> FeatureTagCollection
```

The FeatureTagCollection class represents a collection of FeatureTag objects. See Annex I for the definition of the collection template.

7.8.8 The Contact class – Properties

7.8.8.1

String contactId
The contact identifier represents the public user identity or IMPU used in communication with the contact.

String friendlyName
The friendly name for the user. Used as display name in outgoing messages.

7.8.9 The ContactCollection class

7.8.9.1 General

```
typedef Collection<Contact> ContactCollection
```

The ContactCollection class represents a collection of Contact objects. See Annex I for the definition of the collection template.

In addition to the methods and properties defined for generic collections, the ContactCollection class supports the additional methods defined in 7.8.9.2.

7.8.9.2 Methods

Boolean remove (String contactId)		
Description	Removes the contact represented by contactId from the users contact list. Returns true on success.	
Arguments	contactId	Contact identifier of the user in the contact list.

Boolean add (Contact contact)		
Description	Adds the contact represented by the Contact object to the users contact list. Returns true on success.	
Arguments	contact	Contact object to be added from users contact list.

7.8.10 Extensions to application/oipfCommunicationServices for voice telephony services**7.8.10.1 General**

If an OITF has indicated support for full-duplex Voice Telephony Services functionality by a server by stating <telephony_services>true</telephony_services>, or

<telephony_services video="false">true</telephony_services>, or

<telephony_services video="true">true</telephony_services>

as defined in 9.3.10 in its capability description, the OITF shall support IMS through the use of the following non-visual embedded object:

```
<object type="application/oipfCommunicationServices"/>
```

The full-duplex Voice Telephony Services API provides support for managing the setup and life-cycle of a telephony call session. It also provides the methods to manage the capture devices and the list of preferred codecs to be used.

The full-duplex Voice Telephony Services API may be supported in the combined OITF and IG deployment cases as well as the separated OITF and IG case. It may be supported in other deployment cases. The use of the HNI-IGI interface is optional between the OITF and IG when these are co-deployed.

7.8.10.2 Properties

function **onCallEvent**(Integer eventType, Integer cid, Integer status, String info)

The function that is called when an event related to the identified call is notified.

The specified function is called with four arguments:

- Integer eventType – the type of event. Valid values are:

Value	Description
0	EVENT_INCOMING_CALL: an incoming call is received for the active user
1	EVENT_CALL_PROGRESS: called when the outgoing call is in progress (the request has been received by the remote peer and the local signalling engine is waiting for an answer).
2	EVENT_CALL_RESULT: notifies the result of an outgoing call.
3	EVENT_HANGUP: called when the remote peer hang-up the call.
4	EVENT_SESSION_START: The call session is established and running (media streams can be transmitted and rendered).
5	EVENT_SESSION_END: The call session ended and the related resources are released.
6	EVENT_INCOMING_UPDATE: an incoming update request is received for the active user. ^{a)}
7	EVENT_UPDATE_RESULT: notifies the result of an outgoing update request. ^{a)}
8	EVENT_SESSION_UPDATE: The update of the identified call session is active (e.g.: additional media streams can be transmitted and rendered). ^{a)}
9	EVENT_ERROR: notifies an error event raised during the identified call session.
^{a)} Values not supported for voice only telephony services.	

- Integer cid – call session identifier for the application (Call ID). Call IDs are unique, locally generated positive integer values used to identify a call session.
- Integer status – status information on the event. The content depends on the event
- String info – text field with additional information. The content depends on the event.
- The values of the cid, status and info parameters are defined according to the type of event. Any parameters which are unused for an event shall have the value undefined.
- EVENT_INCOMING_CALL
- cid: call session identifier for the application (Call ID).
- status: call type identifier. Valid Call Type values are:

Value	Description
0	AUDIO_ONLY: full-duplex voice only call
1	VIDEO_ONLY: full-duplex video only call ^{a)}
2	AUDIO_VIDEO: full-duplex video call (voice + video) ^{a)}
^{a)} Values not supported for voice only telephony services	

- info: originating URI. The sender address of the call.
- EVENT_CALL_PROGRESS
- cid: call session identifier for the application (Call ID).
- status: the type of notification coming from the call in progress. This document provides support for a single value; extensions may be defined in future editions.

Value	Description
0	RINGING

- **EVENT_CALL_RESULT**
- **cid**: call session identifier for the application (Call ID).
- **status**: The result of an outgoing call. Valid values are:

Value	Description
0	ACCEPT: The call request has been accepted by the remote peer
1	REFUSE: The call request has been refused by the remote peer
2	TIMEOUT: The call request has been refused due to no response by the remote peer
3	BUSY: The remote peer is currently busy
4	ABORT: a general error occurred

- **info**: if status is equal to 0 (ACCEPT), then the info parameter contains the string representing the value of call type identifier resulting from the negotiation between the peers. Valid call type values are shown in the table below:

Value	Description
0	AUDIO_ONLY: full-duplex voice only call
1	VIDEO_ONLY: full-duplex video only call ^{a)}
2	AUDIO_VIDEO: full-duplex video call (voice + video) ^{a)}
^{a)} Values not supported for voice only telephony services	

- **EVENT_HANGUP**
- **cid**: call session identifier for the application (Call ID).
- **EVENT_SESSION_START**
- **cid**: call session identifier for the application (Call ID).
- **EVENT_SESSION_END**
- **cid**: call session identifier for the application (Call ID).
- **EVENT_INCOMING_UPDATE**
- **cid**: call session identifier for the application (Call ID).
- **status**: call type identifier. Valid call type values are:

Value	Description
0	AUDIO_ONLY: full-duplex voice only call
1	VIDEO_ONLY: full-duplex video only call
2	AUDIO_VIDEO: full-duplex video call (voice + video)

- **EVENT_UPDATE_RESULT**
- **cid**: call session identifier for the application (Call ID).
- **status**: The result of an outgoing call. Valid values are:

Value	Description
0	ACCEPT: The call request has been accepted by the remote peer
1	REFUSE: The call request has been refused by the remote peer
2	TIMEOUT: The call request has been refused due to no response by the remote peer
3	ABORT: a general error occurred

- **EVENT_SESSION_UPDATE**
- **cid**: call session identifier for the application (Call ID).

DeviceInfoCollection getDeviceList(Integer deviceType)								
Description	Returns the list of devices installed on the terminal (or connected) for a specific device type. The device in the first position of the returned list is the default device to be used by the terminal. The position of each device is consistent between method invocations as long as no new devices are connected to the OITF or removed. If an error occurs, the method returns null.							
Arguments	deviceType	Valid types of device are shown in the table below.						
		<table><tr><th>Value</th><th>Description</th></tr><tr><td>0</td><td>Audio Capture devices</td></tr><tr><td>1</td><td>Video Capture devices ^{a)}</td></tr></table>	Value	Description	0	Audio Capture devices	1	Video Capture devices ^{a)}
		Value	Description					
		0	Audio Capture devices					
1	Video Capture devices ^{a)}							
^{a)} Parameters and values not supported for voice only telephony services.								

Boolean setCaptureDevice(Integer deviceType, Integer deviceID)										
Description	Sets the capture device (for a specific device type) that will be used during the call. This method does not affect currently ongoing call sessions. Returns true if the method is successfully executed; false if an error occurred. If the application does not set capture devices (i.e. it does not invoke setCaptureDevice()), then the devices that will be used for the next call session will be the ones in the first position in the DeviceInfoCollection objects returned by the getDeviceList() method for each device type.									
Arguments	deviceType	Valid types of device are shown in the table below.								
		<table><tr><th>Value</th><th>Description</th></tr><tr><td>0</td><td>Audio Capture devices</td></tr><tr><td>1</td><td>Video Capture devices ^{a)}</td></tr><tr><td colspan="2">^{a)} Parameters and values not supported for voice only telephony services.</td></tr></table>	Value	Description	0	Audio Capture devices	1	Video Capture devices ^{a)}	^{a)} Parameters and values not supported for voice only telephony services.	
		Value	Description							
		0	Audio Capture devices							
1	Video Capture devices ^{a)}									
^{a)} Parameters and values not supported for voice only telephony services.										
deviceID	The specific DeviceInfo object id property (in a DeviceInfoCollection) identifying the capture device that will be used for the call.									

CodecInfoCollection getCodecList(Integer streamType)										
Description	Returns the list of codec available on the terminal for a specific stream type. If an error occurs or no codecs are available for the specified stream type, the method returns null .									
Arguments	streamType	Valid stream type values are shown in the table below. <table><tr><th>Value</th><th>Description</th></tr><tr><td>0</td><td>Audio</td></tr><tr><td>1</td><td>Video ^{a)}</td></tr><tr><td colspan="2">a) Parameters and values not supported for voice only telephony services.</td></tr></table>	Value	Description	0	Audio	1	Video ^{a)}	a) Parameters and values not supported for voice only telephony services.	
Value	Description									
0	Audio									
1	Video ^{a)}									
a) Parameters and values not supported for voice only telephony services.										

Boolean setPreferredCodecList (Integer streamType, CodecInfoCollection preferredCodecList)								
Description	Sets a list of preferred codec to be used in the call setup for a specific stream type. Returns true if the method is successfully executed; false if an error occurred. Invocation of this method does not affect currently ongoing call sessions.							
Arguments	streamType	Valid stream type values are shown in the table below.						
		<table><tr><th>Value</th><th>Description</th></tr><tr><td>0</td><td>Audio</td></tr><tr><td>1</td><td>Video ^{a)}</td></tr></table>	Value	Description	0	Audio	1	Video ^{a)}
		Value	Description					
		0	Audio					
1	Video ^{a)}							
^{a)} Parameters and values not supported for voice only telephony services.								
	preferredCodecList	List of codecs to be used during the call setup negotiation (ordered by preference).						

String getCallParameter (Integer cid, String parameter)													
Description	Returns a parameter value for the call session identified by the Cid parameter. This method can be invoked before a call session creation or during an ongoing call session. If the Cid parameter is null , then the retrieved settings will be those that will be applied to the next call sessions that will be created (default for outgoing or incoming call). Returns the value of the parameter or null if an error occurred or the parameter is not supported.												
Arguments	cid	Call session identifier for the application (Call ID).											
	parameter	Mandatory values are shown in the table below. Parameter names are not case sensitive. <table><thead><tr><th>Parameter name</th><th>Description</th></tr></thead><tbody><tr><td>AUDIO_PAUSE</td><td>Audio transmission pause. The parameter can be TRUE or FALSE.</td></tr><tr><td>VIDEO_PAUSE</td><td>Video transmission pause. The parameter can be TRUE or FALSE. ^{a)}</td></tr><tr><td>VIDEO_FPS</td><td>Captured video frame per second ^{a)}</td></tr><tr><td>VIDEO_SIZE</td><td>Captured video size ^{a)} 176 × 144 352 × 288 640 × 480 </td></tr><tr><td>MEDIA_BW</td><td>Audio and video (if available) transmission gross bandwidth (Kbps).</td></tr></tbody></table> ^{a)} Parameters and values not supported for voice only telephony services	Parameter name	Description	AUDIO_PAUSE	Audio transmission pause. The parameter can be TRUE or FALSE .	VIDEO_PAUSE	Video transmission pause. The parameter can be TRUE or FALSE . ^{a)}	VIDEO_FPS	Captured video frame per second ^{a)}	VIDEO_SIZE	Captured video size ^{a)} 176 × 144 352 × 288 640 × 480	MEDIA_BW
Parameter name	Description												
AUDIO_PAUSE	Audio transmission pause. The parameter can be TRUE or FALSE .												
VIDEO_PAUSE	Video transmission pause. The parameter can be TRUE or FALSE . ^{a)}												
VIDEO_FPS	Captured video frame per second ^{a)}												
VIDEO_SIZE	Captured video size ^{a)} 176 × 144 352 × 288 640 × 480												
MEDIA_BW	Audio and video (if available) transmission gross bandwidth (Kbps).												

String setCallParameter(Integer cid, String parameter, String value)																
Description	Sets parameter value for the call session identified by the cid parameter. This method can be invoked before a call session creation or during an ongoing call session. If cid parameter is defined, then the settings will be applied to the call session identified by this Call ID. If cid parameter is null then the settings will be applied at the next call sessions that will be created (default for outgoing or incoming call). Returns true if the method is successfully executed; false if an error occurred.															
Arguments	cid	Call session identifier for the application (Call ID).														
	parameter	Mandatory values are shown in the table below. Parameter names are not case sensitive. <table><tr><th>Parameter name</th><th>Description</th></tr><tr><td>AUDIO_PAUSE</td><td>Audio transmission pause. The parameter can be TRUE or FALSE.</td></tr><tr><td>VIDEO_PAUSE</td><td>Video transmission pause. The parameter can be TRUE or FALSE. ^{a)}</td></tr><tr><td>VIDEO_FPS</td><td>Captured video frame per second ^{a)}</td></tr><tr><td>VIDEO_SIZE</td><td>Captured video size ^{a)} 176 × 144 352 × 288 640 × 480 </td></tr><tr><td>MEDIA_BW</td><td>Audio and video (if available) transmission gross bandwidth (Kbps)</td></tr><tr><td colspan="2">^{a)} Parameters and values not supported for voice only telephony services.</td></tr></table>	Parameter name	Description	AUDIO_PAUSE	Audio transmission pause. The parameter can be TRUE or FALSE.	VIDEO_PAUSE	Video transmission pause. The parameter can be TRUE or FALSE. ^{a)}	VIDEO_FPS	Captured video frame per second ^{a)}	VIDEO_SIZE	Captured video size ^{a)} 176 × 144 352 × 288 640 × 480	MEDIA_BW	Audio and video (if available) transmission gross bandwidth (Kbps)	^{a)} Parameters and values not supported for voice only telephony services.	
	Parameter name	Description														
AUDIO_PAUSE	Audio transmission pause. The parameter can be TRUE or FALSE.															
VIDEO_PAUSE	Video transmission pause. The parameter can be TRUE or FALSE. ^{a)}															
VIDEO_FPS	Captured video frame per second ^{a)}															
VIDEO_SIZE	Captured video size ^{a)} 176 × 144 352 × 288 640 × 480															
MEDIA_BW	Audio and video (if available) transmission gross bandwidth (Kbps)															
^{a)} Parameters and values not supported for voice only telephony services.																
	value	The value for the parameter.														

7.8.10.4 Events

For the intrinsic events listed in the table below, a corresponding DOM event shall be generated in the following manner:

Intrinsic event	Corresponding DOM event	DOM Event properties
onCallEvent	CallEvent	Bubbles: No Cancellable: No Context Info: <code>eventType</code> , <code>cid</code> , <code>status</code> , <code>info</code>

This DOM event is directly dispatched to the event target, and will not bubble nor capture. Applications should not rely on receiving these events during the bubbling or the capturing phase. Applications that use DOM event handlers shall call the `addEventListener()` method on the `application/oipfCommunicationServices` object itself. The third parameter of `addEventListener`, i.e. "useCapture", will be ignored.

7.8.11 Extensions to application/oipfCommunicationServices for video telephony services

7.8.11.1 General

If an OITF has indicated support for full-duplex Video Telephony Services functionality by a server by stating `<telephony_services video="true">true</telephony_services>` as defined in 9.3.9 in its capability description, the OITF shall support communication services through the use of the following non-visual embedded object:

<object type="application/oipfCommunicationServices"/>

The extensions for telephony services provide support for:

- Parameters and values related to video (identified by ^{a)} in previous subclauses).
- Methods to manage the rendering of local and remote video streams through CEA-2014 A/V Control or HTML5 video element.
- Methods to send a session update request and to accept or refuse it. A session update is typically invoked when users want to add video to their currently ongoing voice-only call session.

When a remote or local video is activated on a *target* CEA-2014 A/V Control or HTML5 video element, any currently displayed video content on that object is stopped and released. The activated stream is automatically played.

The full-duplex video telephony services API may be supported in the combined OITF and IG deployment cases as well as the separated OITF and IG case. It may be supported in other deployment cases. The use of the HNI-IGI interface is optional between the OITF and IG when these are co-deployed.

7.8.11.2 Methods

Boolean showRemoteVideo (Integer cid, Integer mode, String idVideoCallObject)								
Description	Activates or deactivates remote peer video rendering. Returns true if the method is successfully executed; false if an error occurred. This method can be invoked as soon as a valid call id is available: after a call method invocation or when an incoming call is notified.							
Arguments	cid	Call session identifier for the application (Call ID).						
	mode	Valid values are shown in the table below. <table><tr><th>Value</th><th>Description</th></tr><tr><td>0</td><td>Deactivates remote video</td></tr><tr><td>1</td><td>Activates remote video</td></tr></table>	Value	Description	0	Deactivates remote video	1	Activates remote video
	Value	Description						
	0	Deactivates remote video						
1	Activates remote video							
idVideoCallObject	ID attribute associated with the HTML tag of the A/V Control object as defined in subclause 7.14 or HTML5 video element in which the video frames will be rendered.							

Boolean showLocalVideoPreview (Integer cid, Integer mode, String idvideoCallObject)								
Description	Activates or deactivates local video preview. This method can be invoked before a call session creation or during an ongoing call session. If cid parameter is defined, then the local video stream will be one currently used in the call session identified by this Call ID. If cid parameter is null then the local video stream will be the one that will be used in the next call session that will be created (outgoing or incoming call). Returns true if the method is successfully executed; false if an error occurred. This method can be invoked before or after the call session setup							
Arguments	cid	Call session identifier for the application (Call ID).						
	mode	Valid values are shown in the table below. <table><tr><th>Value</th><th>Description</th></tr><tr><td>0</td><td>Deactivates local video preview</td></tr><tr><td>1</td><td>Activates local video preview</td></tr></table>	Value	Description	0	Deactivates local video preview	1	Activates local video preview
	Value	Description						
	0	Deactivates local video preview						
1	Activates local video preview							
idVideoCallObject	ID attribute associated with the HTML tag of the A/V Control object as defined in 7.14 or HTML5 video element in which the video frames will be rendered.							

Boolean callUpdate (Integer cid, Integer callType)									
Description	Requests an update for the call session identified by the cid parameter. Returns true if the method is successfully executed; false if an error occurred.								
Arguments	cid	Call session identifier for the application (Call ID).							
	callType	Valid values are shown in the table below.							
	<table><tr><th>Value</th><th>Description</th></tr><tr><td>0</td><td>AUDIO_ONLY: activate a full-duplex voice only call</td></tr><tr><td>1</td><td>VIDEO_ONLY: activate a video only call</td></tr><tr><td>2</td><td>AUDIO_VIDEO: activate a full-duplex video call (voice + video)</td></tr></table>		Value	Description	0	AUDIO_ONLY: activate a full-duplex voice only call	1	VIDEO_ONLY: activate a video only call	2
Value	Description								
0	AUDIO_ONLY: activate a full-duplex voice only call								
1	VIDEO_ONLY: activate a video only call								
2	AUDIO_VIDEO: activate a full-duplex video call (voice + video)								

Boolean callAnswerUpdate(Integer cid, Integer responseUpdate)									
Description	Answers an incoming call. Returns true if the method is successfully executed; false if an error occurred. Note that if an OITF supports full-duplex voice only calls, then the underlying signalling layer shall automatically refuse any update request to video. Requests an update for the call session identified by the cid parameter.								
Arguments	cid	Call session identifier for the application (Call ID).							
	responseUpdate	Valid values are shown in the table below.							
	<table><tr><th>Value</th><th>Description</th></tr><tr><td>0</td><td>UPDATE_ACCEPT: Accepts the update request</td></tr><tr><td>1</td><td>UPDATE_REFUSE: Refuses the update request</td></tr><tr><td>2</td><td>UPDATE_TIMEOUT: Refuses the update request due to no answer from user</td></tr></table>		Value	Description	0	UPDATE_ACCEPT: Accepts the update request	1	UPDATE_REFUSE: Refuses the update request	2
Value	Description								
0	UPDATE_ACCEPT: Accepts the update request								
1	UPDATE_REFUSE: Refuses the update request								
2	UPDATE_TIMEOUT: Refuses the update request due to no answer from user								

7.8.12 The DeviceInfo class

7.8.12.1 General

This class represents a device installed on or connected to the OITF. A device can be for example a capture device, a rendering device etc.

7.8.12.2 Properties

readonly Integer id
A unique, implementation dependent identifier for the capture device defined by the local system. The system shall guarantee that the id assigned to a device will not change during the life of the application

readonly Integer deviceType						
The type of device, valid values are:						
<table><tr><th>Value</th><th>Description</th></tr><tr><td>0</td><td>Audio Capture devices</td></tr><tr><td>1</td><td>Video Capture devices ^{a)}</td></tr></table>	Value	Description	0	Audio Capture devices	1	Video Capture devices ^{a)}
Value	Description					
0	Audio Capture devices					
1	Video Capture devices ^{a)}					
^{a)} Parameters and values not supported for voice only telephony services.						

readonly String deviceName
The friendly name for the capture device. May be used in user messages.

readonly String deviceProductName
The complete name, model number etc. for the capture device.

7.8.13 The DeviceInfoCollection class

```
typedef Collection<DeviceInfo> DeviceInfoCollection
```

The DeviceInfoCollection class represents a collection of DeviceInfo objects. See Annex I for the definition of the collection template.

7.8.14 The CodecInfo class

7.8.14.1 Properties

readonly String codecName
The codec name.

readonly String mimeType
The codec mime-type. A list of possible codec mime-types for multimedia telephony supported by the OIPF Solution are listed in Table 6 of IEC 62766-2-1:2016.

readonly String profile
The codec profile. Normative definition of a subset of standard codec functionalities. The codec profiles for multimedia telephony supported by the OIPF Solution are listed in 6.2.4 of IEC 62766-2-1:2016.

7.8.15 The CodecInfoCollection class

7.8.15.1 General

```
typedef Collection<CodecInfo> CodecInfoCollection
```

The CodecInfoCollection class represents a collection of CodecInfo objects. See Annex I for the definition of the collection template.

In addition to the methods and properties defined for generic collections, the CodecInfoCollection class supports the additional properties and methods defined below.

7.8.15.2 Methods

Boolean moveAt(Integer targetIndex, Integer index)		
Description	Moves a CodecInfo item from the specified index to the specified targetIndex . The operation is performed through an item removal and an insertion of the same item in a new position. The targetIndex is the position in which the element shall be inserted considering position indexes before item removal. During removal and insertion, the other items will shift accordingly. This method shall return true if the operation succeeded, or false if an invalid index was specified (e.g. index > length).	
Arguments	targetIndex	The index in the list to which the item should be moved.
	index	The index in the list of the item that will be moved.

Boolean remove (Integer index)		
Description	Removes the item at the specified index from the CodecInfoCollection . The other items shall shift accordingly. Returns true if the operation succeeded, or false if an invalid index was specified.	
Arguments	index	The index of the item to be removed.

7.9 Parental rating and parental control APIs

7.9.1 General

This subclause defines APIs related to parental ratings and parental control.

Subclauses 7.9.2 through 7.9.4 define a new JavaScript embedded object "application/oipfParentalControlManager" and the related **ParentalRatingScheme** and **ParentalRatingSchemeCollection** objects, which allows applications to construct a new parental rating scheme (and a parental rating value using that scheme), and to temporarily enable or disable viewing of a content item. These APIs shall be supported if an OITF supports parental controls as indicated by value "true" for element <parentalcontrol> (as defined in 9.3.6) in its capability profile.

Subclauses 7.9.5 and 7.9.6 define the **ParentalRating** and **ParentalRatingCollection** objects. These objects are used/referenced by various other objects, such as the **Programme** object as defined in 7.16.2 to indicate a particular parental rating. The support for these objects depends on the support for the subclauses in which these are used.

7.9.2 The application/oipfParentalControlManager embedded object

7.9.2.1 General

If an OITF supports parental controls as indicated by value "true" for element <parentalcontrol> (as defined by 9.3.6) in its capability profile, the OITF shall support the application/oipfParentalControlManager object with the following interface.

The following example shows a possible usage scenario for the application/oipfParentalControlManager, i.e. to add a new parental rating scheme to the parentalRatingSchemes collection:

```
//get a reference to the parental control manager object
var pcManager = document.getElementById("pcmanager");

// add a new rating scheme – in this case, the MPAA rating scheme
pcManager.parentalRatingSchemes.addParentalRatingScheme(
    "urn:mpeg:mpeg7:cs:MPAAParentalRatingCS:2001", "G,PG,PG-13,R,NC-17,NR" );
```

The following example shows a possible usage scenario for the application/oipfParentalControlManager, i.e. to temporarily unblock a blocked content item (e.g. after asking the user to enter the parental control pin):

```
// If a content item is blocked, the event "onParentalRatingChange" can be
// captured, and // the setParentalControlStatus() method can be used to
// temporarily unblock the content // (e.g. after asking the user to enter the
// parental control pin)

function askForPin() { ... }

...

//get a reference to the A/V player object
var avPlayer = document.getElementById("avPlayer");

avPlayer.onParentalRatingChange = function() {
    var pin=askForPin();pcManager.setParentalControlStatus(pin, false);
};
```


7.9.2.2 Properties

readonly ParentalRatingSchemeCollection parentalRatingSchemes
A reference to the collection of rating schemes known by the OITF.

readonly Boolean isPINEntryLocked
The lockout status of the parental control PIN. If the incorrect PIN has been entered too many times in the configured timeout period, parental control PIN entry shall be locked out for a period of time determined by the OITF.

7.9.2.3 Methods

Integer setParentalControlStatus(String pcPIN, Boolean enable)									
Description	As defined in IEC 62766-7, the OITF shall prevent the consumption of a programme when its parental rating doesn't meet the parental rating criterion currently defined in the OITF. Calling this method with enable set to false will temporarily allow the consumption of any blocked programme. Setting the parental control status using this method shall set the status until the consumption of any of all the blocked programmes terminates (e.g. until the content item being played is changed), or another call to the setParentalControlStatus() method is made. Setting the parental control status using this method has the following effect; for the Programme and Channel objects as defined in 7.16.2 and 7.13.12, the blocked property of a programme or channel shall be set to true for programmes whose parental rating does not meet the applicable parental rating criterion, but the locked property shall be set to false. This operation to temporarily disable parental rating control shall be protected by the parental control PIN (i.e. through the pcPIN argument). The return value indicates the success of the operation, and shall take one of the following values: <table border="1"> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>The PIN is correct.</td></tr> <tr> <td>1</td><td>The PIN is incorrect.</td></tr> <tr> <td>2</td><td>PIN entry is locked because an invalid PIN has been entered too many times. The number of invalid PIN attempts before PIN entry is locked is outside the scope of this document.</td></tr> </tbody> </table>	Value	Description	0	The PIN is correct.	1	The PIN is incorrect.	2	PIN entry is locked because an invalid PIN has been entered too many times. The number of invalid PIN attempts before PIN entry is locked is outside the scope of this document.
Value	Description								
0	The PIN is correct.								
1	The PIN is incorrect.								
2	PIN entry is locked because an invalid PIN has been entered too many times. The number of invalid PIN attempts before PIN entry is locked is outside the scope of this document.								
Arguments	<table border="1"> <tr> <td>pcPIN</td><td>The parental control PIN.</td></tr> <tr> <td>enable</td><td>Flag indicating whether parental control should be enabled.</td></tr> </table>	pcPIN	The parental control PIN.	enable	Flag indicating whether parental control should be enabled.				
pcPIN	The parental control PIN.								
enable	Flag indicating whether parental control should be enabled.								

Boolean getParentalControlStatus()	
Description	Returns a flag indicating the temporary parental control status set by setParentalControlStatus() . Note that the returned status covers parental control functionality related to all rating schemes, not only the rating scheme upon which the method is called.

Boolean getBlockUnrated()	
Description	Returns a flag indicating whether or not the OITF has been configured by the user to block content for which a parental rating is absent.

Integer setParentalControlPIN (String oldPcPIN, String newPcPIN)		
Description	Set the parental control PIN.	
	This operation shall be protected by the parental control PIN (if PIN entry is enabled). The return value indicates the success of the operation, and shall take one of the following values:	
	Value	Description
	0	The PIN is correct.
	1	The PIN is incorrect.
	2	PIN entry is locked because an invalid PIN has been entered too many times. The number of invalid PIN attempts before PIN entry is locked is outside the scope of this document.
Arguments	oldPcPIN	The current parental control PIN.
	newPcPIN	The new value for the parental control PIN.

Integer unlockwithParentalControlPIN (String pcPIN, Object target)		
Description	Unlock the object specified by target for viewing if pcPIN contains the correct parental control PIN.	
	The object type of target can be one of the following:	
	• video/broadcast object, in which case the content being presented through this object shall be unlocked until a new channel is selected. • A/V Control object, in which case the content being presented through this object shall be unlocked until a new item of content is played using this object.	
	Otherwise an Invalid Object error shall be returned.	
	The return value indicates the success of the operation, and shall take the following values:	
	Value	Description
	0	The PIN is correct.
	1	The PIN is incorrect.
	2	PIN entry is locked because an invalid PIN has been entered too many times. The number of invalid PIN attempts before PIN entry is locked is outside the scope of this document.
	3	Invalid object.
Arguments	pcPIN	The parental control PIN.
	target	The object to be unlocked.

Integer verifyParentalControlPIN (String pcPIN)		
Description	Verify that the PIN specified by pcPIN is the correct parental control PIN.	
	This method will return one of the following values:	
	Value	Description
	0	The PIN is correct.
	1	The PIN is incorrect.
	2	PIN entry is locked because an invalid PIN has been entered too many times. The number of invalid PIN attempts before PIN entry is locked is outside the scope of this document.
Arguments	pcPIN	The parental control PIN to be verified.

Integer setBlockUnrated(String pcPIN, Boolean block)		
Description	Set whether programmes for which no parental rating has been retrieved from the metadata client nor defined by the service provider should be blocked automatically by the terminal.	
	This operation shall be protected by the parental control PIN (if PIN entry is enabled). The return value indicates the success of the operation, and shall take one of the following values:	
	Value	Description
	0	The PIN is correct.
	1	The PIN is incorrect.
Arguments	2	PIN entry is locked because an invalid PIN has been entered too many times. The number of invalid PIN attempts before PIN entry is locked is outside the scope of this document.
	pcPIN	The parental control PIN.
	block	Flag indicating whether programmes shall be blocked.

7.9.3 The ParentalRatingScheme class

7.9.3.1 General

```
typedef Collection<String> ParentalRatingScheme
```

A ParentalRatingScheme describes a single parental rating scheme that may be in use for rating content, for example the MPAA or BBFC rating schemes. It is a collection of strings representing rating values, which next to the properties and methods defined below shall support the array notation to access the rating values in this collection. For the natively OITF supported parental rating systems, the values shall be ordered by the OITF to allow the rating values to be compared in the manner as defined for property threshold for the respective parental rating system. Using a threshold as defined in this API may not necessarily be the proper way in which parental rating filtering is applied on the OITF; for example, the US FCC requirements take precedence for device to be imported to the US.

The parental rating schemes supported by a receiver may vary between deployments.

See Annex I for the definition of the collection template. In addition to the methods and properties defined for generic collections, the ParentalRatingScheme class supports the additional properties and methods defined below.

7.9.3.2 Properties

readonly String name
The unique name that identifies the parental rating scheme. Valid strings include: - the URI of one of the MPEG-7 classification schemes representing a parental rating scheme as defined by the "uri" attribute of one of the parental rating <ClassificationScheme> elements in ISO/IEC 15938-5:2003. - the string value "urn:oipf:GermanyFSKCS" to represent the GermanyFSK rating scheme as defined in IEC 62766-3. - the string value "dvb-si": this means that the scheme of a minimum recommended age encoded as per ratings 0x01 to 0x0f in the parental rating descriptor from ETSI EN 300 468, is used to represent the parental rating values. - note that if the broadcaster defined range from 0x10 to 0xff is used then that would be a different parental rating scheme and not "dvb-si". If the value of "name" is "dvb-si", the ParentalRatingScheme remains empty (i.e. ParentalRatingScheme.length == 0).

readonly ParentalRating threshold
The parental rating threshold that is currently in use by the OITF's parental control system for this rating scheme, which is encoded as a ParentalRating object in the following manner: If the value of the "name" property of the ParentalRatingScheme object is unequal to "dvb-si", then: - the "value" property of the threshold object represents the value for which items with a ParentalRating.value greater or equal to the "value" property of the threshold object have been configured by the OITF's parental control subsystem to be blocked. - the "labels" property of the threshold object represents the bit map of zero or more flags for which items with a ParentalRating.labels property with any of the same flags set have been configured by the OITF's parental control subsystem to be blocked. If the value of the name property of the ParentalRatingScheme object is "dvb-si", the threshold indicates a minimum recommended age encoded as per ETSI EN 300 468 at which or above which the content is being blocked by the OITF's parental control subsystem. Note that the value property as an index into the ParentalRating object that defines the threshold can be 1 larger than the value of ParentalRatingScheme.length to convey that no content is being blocked by the parental control subsystem.

7.9.3.3 Methods

Integer indexOf(String ratingValue)		
Description	Return the index of the rating represented by attribute ratingValue inside the parental rating scheme string collection, or -1 if the rating value cannot be found in the collection.	
Arguments	ratingValue	The string representation of a parental rating value. See property name in 7.9.2.2 for more information about possible values. Values are not case sensitive.

String iconUri(Integer index)		
Description	Return the URI of the icon representing the rating at index in the rating scheme, or undefined if no item is present at that position. If no icon is available, this method shall return null .	
Arguments	index	The index of the parental rating scheme.

7.9.4 The ParentalRatingSchemeCollection class

7.9.4.1 General

```
typedef Collection<ParentalRatingScheme> ParentalRatingSchemeCollection
```

A **ParentalRatingSchemeCollection** represents a collection of parental rating schemes, e.g. as returned by property **parentalRatingSchemes** of the "application/oipfParentalControlManager" object as defined in 7.9.1. Next to the properties and methods defined in 7.9.4.2, a **ParentalRatingSchemeCollection** object shall support the array notation to access the parental rating scheme objects in this collection.

See Annex I for the definition of the collection template. In addition to the methods and properties defined for generic collections, the **ParentalRatingSchemeCollection** class supports the additional properties and methods defined below.

7.9.4.2 Methods

ParentalRatingScheme addParentalRatingScheme (String name, String values)		
Description	Create a new ParentalRatingScheme object and adds it to the ParentalRatingSchemeCollection. Applications may use this method to register additional parental rating schemes with the platform. When registered, the new parental rating scheme shall (temporarily) be accessible through the parentalRatingSchemes property of the "application/oipfParentalControlmanager" object as defined in 7.9.2. The application shall make sure that the values are ordered in such a way to allow the rating values to be compared in the manner as defined for the threshold property for the respective parental rating system. This method returns a reference to the ParentalRatingScheme object representing the added scheme. If the value of the name parameter corresponds to an already-registered rating scheme, this method returns a reference to the existing ParentalRatingScheme object. If the newly defined rating scheme was not known to the OITF, the scheme may be stored persistently, and the OITF may offer a UI to set the parental rating blocking criteria for the newly added parental rating scheme. If the OITF has successfully stored (persistently or not persistently) the additional parental rating scheme, the method shall return a non-null ParentalRatingScheme object.	
Arguments	name	A unique string identifying the parental rating scheme to which this value refers. See property name in 7.9.2.2 for more information about possible values.
	values	A comma-separated list of the possible values in the rating scheme, in ascending order of severity. In case the rating scheme is one of the ISO/IEC 15938-5:2003 rating classification schemes, this means that the list of parental rating values contains the values as specified by the <Name> elements of the <Term> elements in the order of appearance as they are defined for the classification scheme, with the exception of the Internet Content Rating Association (ICRA) based ratings, for which the reverse order has to be applied. The values shall be ordered in such a way to allow the rating values to be compared in the manner as defined for property threshold for the respective parental rating system.

ParentalRatingScheme getParentalRatingScheme (String name)		
Description	This method returns a reference to the ParentalRatingScheme object that is associated with the given scheme as specified through parameter "name". If the value of name does not correspond to the name property of any of the ParentalRatingScheme objects in the ParentalRatingSchemeCollection, the method shall return undefined.	
Arguments	name	The unique name identifying a parental rating scheme.

7.9.5 The ParentalRating class

7.9.5.1 General

A **ParentalRating** object describes a parental rating value for a programme or channel. The **ParentalRating** object identifies both the rating scheme in use, and the parental rating value within that scheme.

In case of a BCG, the values of the properties in this object will be read from the **ParentalGuidance** element that is the child of a programme's BCG description.

Example usage:

```
<!-- This example shows a possible usage scenario for the ParentalRating
      data structure, i.e. to create a new programme to record and set
      parental rating to MPAA parental rating to PG-13.
-->
...
<script type="text/javascript" language="Javascript1.5">
```

```
// get a reference to the recorder object
var recorder = document.getElementById("recorder");

// create new programme to record
var myProgramme = recorder.createProgrammeObject();

// add a new parental rating value to myProgramme, in this case the
// programme is rated PG-13 for the US using the MPAA Parental rating scheme.
myProgramme.parentalRatings.addParentalRating(
    "urn:mpeg:mpeg7:cs:MPAAParentalRatingCS:2001", "PG-13", 2, 0, "US"
);

</script>
...
<object id="recorder" type="application/oipfRecordingScheduler"/>
```

7.9.5.2 Properties

readonly String name
The string representation of the parental rating value for the respective rating scheme denoted by property scheme. Valid strings include: - if the value of property scheme represents one of the parental rating classification scheme names identified by ISO/IEC 15938-5:2003. The string representation of one of the parental rating values as defined by one of the <Name> elements. - if the value of property scheme is "urn:oipf:GermanyFSKCS", the string representation of one the values for the Germany FSK rating scheme as defined in IEC 62766-3. - if the value of property scheme is equal to "dvb-si", this means that the scheme of a minimum recommended age encoded as per ratings 0x01 to 0x0f in the parental rating descriptor from ETSI EN 300 468, which corresponds to rating_type 0 in IEC 62766-3. NOTE If the broadcaster defined range from 0x10 to 0xff is used, then that would be a different parental rating scheme and not "dvb-si". An example of a valid parental rating value is "PG-13".

readonly String scheme
Unique name identifying the parental rating guidance scheme to which this parental rating value refers. Valid strings include: - the URI of one of the MPEG-7 classification schemes representing a parental rating scheme as defined by the "uri" attribute of one of the parental rating <ClassificationScheme> elements in ISO/IEC 15938-5:2003. - the string value "urn:oipf:GermanyFSKCS" to represent the Germany FSK rating scheme as defined in IEC 62766-3. - the string value "dvb-si": this means that the scheme of a minimum recommended age encoded as per ETSI EN 300 468 is used to represent the parental rating values.

readonly Integer value
The parental rating value represented as an index into the set of values defined as part of the ParentalRatingScheme identified through property "scheme". If an associated ParentalRatingScheme object can be found by calling method getParentalRatingScheme() on property parentalRatingSchemes of the application/oipfParentalControlManager object and the value of property scheme is not equal to "dvb-si", then the value property shall represent the index of the parental rating value inside the ParentalRatingScheme object, or -1 if the value cannot be found. If the value of property scheme is equal to "dvb-si", then this property shall be the integer representation of the string value of ParentalRating property name. If no associated ParentalRatingScheme object can be found by calling method getParentalRatingScheme on property parentalRatingSchemes of the application/oipfParentalControlManager object, then the value property shall have value undefined.

readonly Integer labels	
The labels property represents a set of parental advisory flags that may provide additional information about the rating.	
The value of this field is a 32 bit integer value that represents a binary mask corresponding to the sum of zero or more label values defined in the table below. If no labels have been explicitly set, the value for the "labels" property shall be 0.	
Valid labels include:	
Value	Description
1	Indicates that a content item features sexually suggestive dialogue.
2	Indicates that a content item features strong language.
4	Indicates that a content item features sexual situations.
8	Indicates that a content item features violence.
16	Indicates that a content item features fantasy violence.
32	Indicates that a content item features disturbing scenes.
64	Indicates that a content item features portrayals of discrimination.
128	Indicates that a content item features scenes of illegal drug use.
256	Indicates that a content item features strobing that could impact viewers suffering from photosensitive epilepsy.

readonly String region
The region to which the parental rating value applies as an alpha-2 region code as defined in ISO 3166-1. Returns undefined if no specific region has been defined.

7.9.6 The ParentalRatingCollection class

7.9.6.1 General

```
typedef Collection<ParentalRating> ParentalRatingCollection
```

A `ParentalRatingCollection` represents a collection of parental rating values. See Annex I for the definition of the collection template.

In addition to the methods and properties defined for generic collections, the `ParentalRatingCollection` class supports the additional properties and methods defined below.

7.9.6.2 Methods

void addParentalRating (String scheme, String name, Integer value, Integer labels, String region)		
Description	Creates a ParentalRating object instance for a given parental rating scheme and parental rating value, and adds it to the ParentalRatingCollection for a programme or channel.	
Arguments	scheme	A unique string identifying the parental rating scheme to which this value refers. See property scheme in 7.9.5.2 for more information about possible values.
	name	A string representation of the parental rating value. See property name in 7.9.5.2 for more information about possible values. Values are not case sensitive.
	value	The parental rating value represented as an Integer. See property value in 7.9.5.2 for more information about possible values.
	labels	A set of content rating labels that may provide additional information about the rating. See property labels in 7.9.5.2 for more information about possible values.
	region	The region to which the parental rating value applies as an alpha-2 region code as defined in ISO 3166-1. The value of this argument shall be null or undefined if no specific region has been identified. Values are not case sensitive.

7.10 Scheduled Recording APIs

7.10.1 General

Subclause 7.10 describes the APIs needed to support control by a DAE application of the recording (PVR) functionality available to an OITF, including time-shift support, scheduled recording and storage of basic metadata about recorded items. Changes made by a DAE application to properties that can also be set by the OITF may be overwritten by the OITF from metadata without warning.

Subclause 7.10 shall apply for OITFs that have indicated <recording> with value "true" as defined in 9.3.4 in its capability description.

7.10.2 The application/oipfRecordingScheduler embedded object

7.10.2.1 General

The OITF shall support the scheduling of recordings of broadcasts through the use of the following non-visual embedded object:

```
<object type="application/oipfRecordingsScheduler"/>
```

Note that the functionality in this subclause shall adhere to the security model as specified in 10.1.

Which channels can be recorded shall be indicated by the **ipBroadcast**, **DASH** and **HAS** attributes in the PVR capability indication (see 9.3.4). Within the channels indicated by these attributes, recording of both channels stored in the channel list and locally defined channels shall be supported.

7.10.2.2 Methods

ScheduledRecording record (Programme programme)		
Description	Requests the scheduler to schedule the recording of the programme identified by the programmeID property of the programme. If the programmeIDType of the programme has the value ID_TVA_GROUP_CRID then the ScheduledRecording object returned by this method shall be a "parent" scheduled recording object that conceptually represents the recording. Each individual programme in the shall be represented by a separate ScheduledRecording object. Note that ScheduledRecording objects for individual programmes may not be created until the CRID has been partially or completely resolved. The start time, duration and other properties of the programme shall not be used for scheduling any recording. Individual programmes shall be recorded if any entries in a programme's associated groupCRIDS collection matches the group CRID specified in the programmeID property of any "parent" recording. The other data contained in the programme object is used solely for annotation of the (scheduled) recording. If such programme metadata is provided, it shall be retained in the ScheduledRecording object that is returned if the recording of the programme was scheduled successfully, reflecting the possibility that not all relevant metadata might be available to the scheduler. When the programme is recorded, the metadata in the associated Recording object shall be updated with the metadata from the broadcast stream if such metadata is available. If the recording could not be scheduled due to a scheduling conflict or lack of resources the value null is returned. Note that the actual implementation of this method should enable the scheduler to identify the domain of the service that issues the scheduling request in order to support future retrieval of the scheduled recording through the getScheduledRecordings method.	
Arguments	programme	The programme to be recorded, as defined in 7.16.2.

IECNORM.COM : Click to view the full PDF of IEC 62766-5-1:2017

ScheduledRecording recordAt(Integer startTime, Integer duration, Integer repeatDays, String channelId)																									
Description	Requests the scheduler to schedule the recording of the broadcast to be received over the channel identified by channelID, starting at startTime and stopping at startTime + duration. If the recording was scheduled successfully, the resulting ScheduledRecording object is returned. If the recording could not be scheduled due to a scheduling conflict or lack of resources, the value null is returned. The OITF should associate metadata with recordings scheduled using this method. This metadata may be obtained from the broadcast being recorded (for example DVB-SI in an MPEG-2 transport stream) or from other sources of metadata. If an application anticipates that the OITF may not be able to obtain any metadata, it should instead of using this method; • create a Programme object (using the createProgramme() method) containing the channelID, startTime and duration • populate that Programme object with metadata • pass that Programme object to the record(Programme) method. Note that the actual implementation of this method should enable the scheduler to identify the domain of the service that issues the scheduling request in order to support future retrieval of the scheduled recording through the getScheduledRecordings() method.																								
	Arguments	<table><tr><td>startTime</td><td>The start of the time period of the recording measured in seconds since midnight (GMT) on 1/1/1970. If the start time occurs in the past and the current time is within the specified duration of the recording, the OITF shall start recording immediately and may record any earlier content from the current programme that is available (e.g. stored in a time-shift buffer).</td></tr><tr><td>duration</td><td>The duration of the recording in seconds.</td></tr><tr><td>repeatDays</td><td> Bitfield indicating which days of the week the recording should be repeated. Values are as follows: <table><tr><th>Day</th><th>Bitfield Value</th></tr><tr><td>Sunday</td><td>0x01 (i.e. 00000001)</td></tr><tr><td>Monday</td><td>0x02 (i.e. 00000010)</td></tr><tr><td>Tuesday</td><td>0x04 (i.e. 00000100)</td></tr><tr><td>Wednesday</td><td>0x08 (i.e. 00001000)</td></tr><tr><td>Thursday</td><td>0x10 (i.e. 00010000)</td></tr><tr><td>Friday</td><td>0x20 (i.e. 00100000)</td></tr><tr><td>Saturday</td><td>0x40 (i.e. 01000000)</td></tr></table> These bitfield values can be 'OR'-ed together to repeat a recording on more than one day of a week (e.g. weekdays) A value of 0x00 indicates that the recording will not be repeated. </td></tr><tr><td>channelID</td><td>The identifier of the channel from which the broadcasted content is to be recorded. Specifies either a ccid or ipBroadcastID (as defined by the Channel object in 7.13.12)</td></tr></table>	startTime	The start of the time period of the recording measured in seconds since midnight (GMT) on 1/1/1970. If the start time occurs in the past and the current time is within the specified duration of the recording, the OITF shall start recording immediately and may record any earlier content from the current programme that is available (e.g. stored in a time-shift buffer).	duration	The duration of the recording in seconds.	repeatDays	Bitfield indicating which days of the week the recording should be repeated. Values are as follows: <table><tr><th>Day</th><th>Bitfield Value</th></tr><tr><td>Sunday</td><td>0x01 (i.e. 00000001)</td></tr><tr><td>Monday</td><td>0x02 (i.e. 00000010)</td></tr><tr><td>Tuesday</td><td>0x04 (i.e. 00000100)</td></tr><tr><td>Wednesday</td><td>0x08 (i.e. 00001000)</td></tr><tr><td>Thursday</td><td>0x10 (i.e. 00010000)</td></tr><tr><td>Friday</td><td>0x20 (i.e. 00100000)</td></tr><tr><td>Saturday</td><td>0x40 (i.e. 01000000)</td></tr></table> These bitfield values can be 'OR'-ed together to repeat a recording on more than one day of a week (e.g. weekdays) A value of 0x00 indicates that the recording will not be repeated.	Day	Bitfield Value	Sunday	0x01 (i.e. 00000001)	Monday	0x02 (i.e. 00000010)	Tuesday	0x04 (i.e. 00000100)	Wednesday	0x08 (i.e. 00001000)	Thursday	0x10 (i.e. 00010000)	Friday	0x20 (i.e. 00100000)	Saturday	0x40 (i.e. 01000000)	channelID
startTime	The start of the time period of the recording measured in seconds since midnight (GMT) on 1/1/1970. If the start time occurs in the past and the current time is within the specified duration of the recording, the OITF shall start recording immediately and may record any earlier content from the current programme that is available (e.g. stored in a time-shift buffer).																								
duration	The duration of the recording in seconds.																								
repeatDays	Bitfield indicating which days of the week the recording should be repeated. Values are as follows: <table><tr><th>Day</th><th>Bitfield Value</th></tr><tr><td>Sunday</td><td>0x01 (i.e. 00000001)</td></tr><tr><td>Monday</td><td>0x02 (i.e. 00000010)</td></tr><tr><td>Tuesday</td><td>0x04 (i.e. 00000100)</td></tr><tr><td>Wednesday</td><td>0x08 (i.e. 00001000)</td></tr><tr><td>Thursday</td><td>0x10 (i.e. 00010000)</td></tr><tr><td>Friday</td><td>0x20 (i.e. 00100000)</td></tr><tr><td>Saturday</td><td>0x40 (i.e. 01000000)</td></tr></table> These bitfield values can be 'OR'-ed together to repeat a recording on more than one day of a week (e.g. weekdays) A value of 0x00 indicates that the recording will not be repeated.	Day	Bitfield Value	Sunday	0x01 (i.e. 00000001)	Monday	0x02 (i.e. 00000010)	Tuesday	0x04 (i.e. 00000100)	Wednesday	0x08 (i.e. 00001000)	Thursday	0x10 (i.e. 00010000)	Friday	0x20 (i.e. 00100000)	Saturday	0x40 (i.e. 01000000)								
Day	Bitfield Value																								
Sunday	0x01 (i.e. 00000001)																								
Monday	0x02 (i.e. 00000010)																								
Tuesday	0x04 (i.e. 00000100)																								
Wednesday	0x08 (i.e. 00001000)																								
Thursday	0x10 (i.e. 00010000)																								
Friday	0x20 (i.e. 00100000)																								
Saturday	0x40 (i.e. 01000000)																								
channelID	The identifier of the channel from which the broadcasted content is to be recorded. Specifies either a ccid or ipBroadcastID (as defined by the Channel object in 7.13.12)																								

ScheduledRecordingCollection getScheduledRecordings()	
Description	Returns a subset of all the recordings that are scheduled but which have not yet started. The subset shall include only scheduled recordings that were scheduled using a service from the same FQDN as the domain of the service that calls the method.

ChannelConfig getChannelConfig()	
Description	Returns the channel line-up of the OITF in the form of a ChannelConfig object as defined in 7.13.10. The ChannelConfig object returned from this function shall be identical to the ChannelConfig object returned from the getChannelConfig() method on the video/broadcast object as defined in 7.13.2.4.

void remove(ScheduledRecording recording)		
Description	Remove a recording (either scheduled, in-progress or completed). For non-privileged applications, recordings shall only be removed when they are scheduled but not yet started and the recording was scheduled by the current service. As with the record() method, only the programmeID property of the scheduled recording shall be used to identify the scheduled recording to remove where this property is available. The other data contained in the scheduled recording shall not be used when removing a recording scheduled using methods other than recordAt(). For recordings scheduled using recordAt(), the data used to identify the recording to remove is implementation dependent. If the programmeIDType property has the value ID_TVA_GROUP_CRID then the OITF shall cancel the recording of the specified group. If an A/V Control object is presenting the indicated recording, then the state of the A/V Control object shall be automatically changed to 6 (the error state).	
Arguments	recording	The scheduled recording to be removed.

Programme createProgrammeObject()	
Description	Factory method to create an instance of Programme.

7.10.3 The ScheduledRecording class

7.10.3.1 General

The **ScheduledRecording** object represents a scheduled programme in the system, i.e. a recording that is scheduled but which has not yet started. For group recordings (e.g. recording an entire series), a **ScheduledRecording** object is also used to represent a "parent" recording that enables management of the group recording without representing any of the actual recordings in the group. The values of the properties of a **ScheduledRecording** (except for **startPadding** and **endPadding**) are provided when the object is created using one of the **record()** methods in 7.10.2, for example by using a corresponding **Programme** object as argument for the **record()** method, and can not be changed for this scheduled recording object (except for **startPadding** and **endPadding**).

7.10.3.2 Constants

The following table lists the constants for recording states.

Name	Use
RECORDING_SCHEDULED	Recording has been newly scheduled.
RECORDING_REC_STARTED	Recording has started.
RECORDING_REC_COMPLETED	Recording has successfully completed.
RECORDING_REC_PARTIALLY_COMPLETED	The recording has only partially completed due to insufficient storage space, a clash or hardware failure. There are three possible conditions for this: 1) the end of the recording is missed; 2) the start of the recording is missed; 3) a piece from the centre of the recording is missed (e.g. due to the receiver rebooting or a transient failure of the network connection).
RECORDING_ERROR	An error has been encountered. Refer to detailed error codes for details on the error.

This document does not define values for these constants. Implementations may use any values as long as the value of each constant is unique.

The following table lists the constants for detailed error codes when a recording failed to complete.

Name	Use
ERROR_REC_RESOURCE_LIMITATION	The recording sub-system is unable to record due to resource limitations.
ERROR_INSUFFICIENT_STORAGE	There is insufficient storage space available (some of the recording may be available).
ERROR_REC_UNKNOWN	Recording has stopped before completion due to unknown (probably hardware) failure.

This document does not define values for these constants. Implementations may use any values as long as the value of each constant is unique.

The following table lists the constants for programme ID types.

Name	Value	Use
ID_TVA_CRID	0	Used in the <code>programmeIDType</code> property to indicate that the programme is identified by its TV-Anytime CRID (Content Reference Identifier).
ID_DVB_EVENT	1	Used in the <code>programmeIDType</code> property to indicate that the programme is identified by a DVB URL referencing a DVB-SI event as enabled by 5.2.4 of IEC 62766-3:2016. Support for this constant is optional.
ID_TVA_GROUP_CRID	2	Used in the <code>programmeIDType</code> property to indicate that the Programme object represents a group of programmes identified by a TV-Anytime group CRID.

7.10.3.3 Properties

readonly Integer state
The state of the recording. Valid values are: RECORDING_REC_STARTED RECORDING_REC_COMPLETED RECORDING_REC_PARTIALLY_COMPLETED RECORDING_SCHEDULED RECORDING_ERROR

readonly Integer error
If the state of the recording has changed due to an error, this field contains an error code detailing the type of error. This is only valid if the value of the state argument is <code>RECORDING_ERROR</code> or <code>RECORDING_REC_PARTIALLY_COMPLETED</code> otherwise this property shall be <code>null</code> . Valid values are: ERROR_REC_RESOURCE_LIMITATION ERROR_INSUFFICIENT_STORAGE ERROR_REC_UNKNOWN

readonly String scheduleID
An identifier for this scheduled recording. This value shall be unique to this scheduled recording. For a recording object this identifier can be used to associate which scheduled recording object this recording was created from.

String **customID**

An identifier for this scheduled recording. This value is an identifier that the DAE application can set in order to keep track of scheduled recordings. It is not changed by the OITF.

Integer **startPadding**

The amount of padding to add at the start of a scheduled recording, in seconds. If the value of this property is **undefined**, an OITF defined start padding will be used. The default OITF defined start padding may be changed through property **pvrStartPadding** of the **Configuration** class as defined in 7.3.3. When a recording is due to start, the OITF may use a smaller amount of padding in order to avoid conflicts with other recordings.

Positive values of this property shall cause the recording to start earlier than the specified start time (i.e. the actual duration of the recording shall be increased); negative values shall cause the recording to start later than the specified start time (i.e. the actual duration of the recording shall be decreased).

Integer **endPadding**

The amount of padding to add at the end of a scheduled recording, in seconds. If the value of this property is **undefined**, an OITF defined end padding will be used. The default OITF defined end padding may be changed through property **pvrEndPadding** of the **Configuration** class as defined in 7.3.3. When a recording is in progress, the OITF may use a smaller amount of padding in order to avoid conflicts with other recordings.

Positive values of this property shall cause the recording to end later than the specified end time (i.e. the actual duration of the recording shall be increased); negative values shall cause the recording to end earlier than the specified end time (i.e. the actual duration of the recording shall be decreased).

readonly Integer **repeatDays**

Bitfield indicating which days of the week the recording should be repeated. Values are as follows:

Day	Bitfield Value
Sunday	0x01 (i.e. 00000001)
Monday	0x02 (i.e. 00000010)
Tuesday	0x04 (i.e. 00000100)
Wednesday	0x08 (i.e. 00001000)
Thursday	0x10 (i.e. 00010000)
Friday	0x20 (i.e. 00100000)
Saturday	0x40 (i.e. 01000000)

These bitfield values can be arithmetically summed to repeat a recording on more than one day of a week (e.g. weekdays)

A value of 0x00 indicates that the recording will not be repeated.

String **name**

The short name of the scheduled recording, for example 'Star Trek: DS9'.

String **longName**

The long name of the scheduled recording, for example 'Star Trek: Deep Space Nine'. If the long name is not available, this property will be **undefined**.

String **description**

The description of the scheduled recording, for example an episode synopsis. If no description is available, this property will be **undefined**.

String longDescription

The long description of the programme. If no description is available, this property will be **undefined**.

readonly Integer startTime

The start time of the scheduled recording, measured in seconds since midnight (GMT) on 1/1/1970. The value for the **startPadding** property can be used to indicate if the recording has to be started before the **startTime** (as defined by the **Programme** class).

readonly Integer duration

The duration of the scheduled recording (in seconds). The value for the **endPadding** property can be used to indicate how long the recording has to be continued after the specified duration of the recording.

readonly Channel channel

Reference to the broadcast channel where the scheduled programme is available.

readonly Boolean isManual

true if the recording was scheduled using **oipfRecordingsScheduler.recordAt()** or using a terminal-specific approach that does not use guide data to determine what to record, **false** otherwise.

If **false**, then any fields whose name matches a field in the **Programme** object contains details from the programme guide on the programme that has been recorded. If **true**, only the **channel**, **startTime** and **duration** properties are required to be valid.

readonly String programmeID

The unique identifier of the scheduled programme or series, e.g. a TV-Anytime CRID (Content Reference Identifier). For recordings scheduled using the **oipfRecordingsScheduler.recordAt()** method, the value of this property may be undefined.

readonly Integer programmeIDType

The type of identification used to reference the programme, as indicated by one of the **ID_*** constants defined in 7.10.3.2. For recordings scheduled using the **oipfRecordingsScheduler.recordAt()** method, the value of this property may be undefined.

readonly Integer episode

The episode number for the programme if it is part of a series. This property is **undefined** when the programme is not part of a series or the information is not available.

readonly Integer totalEpisodes

If the programme is part of a series, the total number of episodes in the series. This property is **undefined** when the programme is not part of a series or the information is not available.

readonly ParentalRatingCollection parentalRatings
A collection of parental rating values for the programme for zero or more parental rating schemes supported by the OITF. The value of this property is typically provided by a corresponding "Programme" object that is used to schedule the recording and can not be changed for this scheduled recording object. If no parental rating information is available for this scheduled recording, this property is a ParentalRatingCollection object (as defined in 7.9.6) with length 0. Note that if the parentalRating property contains a certain parental rating (e.g. PG-13) and the broadcast channel associated with this scheduled recording has metadata that says that the content is rated PG-16, then the conflict resolution is implementation dependent. Note that this property was formerly called "parentalRating" (singular not plural).

String customMetadata
Application-specific information for this recording. This value is information that the DAE application can set in order to retain additional information on this scheduled recording. It is not changed by the OITF. The OITF shall support values up to and including 16 KB in size. Strings longer than this may get truncated.

7.10.4 The ScheduledRecordingCollection class

```
typedef Collection<ScheduledRecording> ScheduledRecordingCollection
```

The ScheduledRecordingCollection class represents a collection of ScheduledRecording objects. See Annex I for the definition of the collection template.

7.10.5 Extension to application/oipfRecordingsScheduler for control of recordings

7.10.5.1 General

The OITF shall support the following extensions to the application/oipfRecordingsScheduler object defined in 7.10.2.

This subsubclause shall apply for OITFs that have indicated support for the extended PVR management functionality by adding the attribute 'manageRecordings = true' to the <recording> element in the client capability description as defined in 9.3.4.

The functionality as described in this subclause is subject to the security model of Clause 10.

7.10.5.2 Properties

readonly ScheduledRecordingCollection recordings
Provides a list of scheduled and recorded programmes in the system. This property may only provide access to a subset of the full list of recordings, as determined by the value of the manageRecordings attribute of the <recording> element in the client capability description (see 9.3.4).

readonly DiscInfo discInfo
Get information about the status of the local storage device. The DiscInfo class is defined in 7.16.4.

function **onPVREvent**(Integer state, ScheduledRecording recording)

This function is the DOM 0 event handler for notification of changes in the state of recordings. The specified function is called with the following arguments:

- **Integer state** – The current state of the recording. One of:

Value	Description
1	The recording has started.
2	The recording has stopped, having completed.
3	The recording sub-system is unable to record due to resource limitations.
4	There is insufficient storage space available (some of the recording may be available).
6	The recording has stopped before completion due to unknown (probably hardware) failure.
7	The recording has been newly scheduled.
8	The recording has been deleted (for complete or in-progress recordings) or removed from the schedule (for scheduled recordings).
9	The recording is due to start in a short time.
10	The recording has been updated. For performance reasons, OITFs should not dispatch events with the state when only the duration of an in-progress recording has changed.

- **ScheduledRecording recording** – The recording to which this event refers.

7.10.5.3 Methods

Recording getRecording (String id)		
Description	Returns the Recording object for which the value of the Recording.id property corresponds to the given id parameter. If such a Recording does not exist, the method returns null .	
Arguments	id	Identifier corresponding to the id property of a Recording object.

void stop (Recording recording)		
Description	Stop an in-progress recording. The recording shall not be deleted.	
Arguments	recording	The recording to be stopped.

void refresh ()	
Description	Update the recordings property to show the current status of all recordings.

Boolean update (String id, Integer startTime, Integer duration, Integer repeatDays)		
Description	Requests the scheduler to update a scheduled or ongoing recording. For scheduled recordings the properties startTime, duration and repeatDays can be modified. For ongoing recordings only the duration property may be modified. This method shall return true if the operation succeeded, or false if for any reason it rescheduling is not possible (e.g. the updated recording overlaps with another scheduled recording and there are insufficient system resources to do both). If the method returns false, then no changes shall be made to the recording.	
Arguments	id	The id of the recording to update
	startTime	The new start time of the recording, or undefined if the start time is not to be updated.
	duration	The new duration of the recording, or undefined if the duration is not to be updated.
	repeatDays	The new set of days on which the recording is to be repeated, or undefined if this is not to be updated.

7.10.5.4 Events

For the intrinsic events listed in the table below, a corresponding DOM event shall be generated in the following manner:

Intrinsic event	Corresponding DOM event	DOM Event properties
onPVREvent	PVREvent	Bubbles: No Cancellable: No Context Info: state , recording

The DOM events are directly dispatched to the event target, and will not bubble nor capture. Remote UIs should not rely on receiving these events during the bubbling or the capturing phase. Remote UIs that use DOM event handlers shall call the `addEventListener()` method on the `application/oipfscheduledRecording` object itself. The third parameter of `addEventListener`, i.e. "useCapture", will be ignored.

7.10.6 The Recording class

7.10.6.1 General

The `Recording` class represents an in-progress or completed recording being made available through the extended PVR management functionality as defined in 7.10.5. Recordings for which no data has yet been recorded are represented by the `ScheduledRecording` class.

This class implements the `ScheduledRecording` interface (see 7.10.3) to provide access to the information relating to the scheduling of the recording. The difference between scheduled recordings, in-progress recordings and completed recordings is primarily what properties are populated with values. In addition, for recorded and in-progress recordings the following is true:

- the `startPadding` property is read only;
- for in-progress recordings, changes to the value of the `endPadding` property shall modify the actual duration of the recording. If the value of the `endPadding` property is changed so that the current actual duration of the recording exceeds the new actual duration specified by the sum of the `startPadding`, `duration` and `endPadding` properties, the

recording shall be stopped immediately. Changing the value of this property for a completed recording shall have no effect.

Recordings may be "manual" in that they simply record a channel at a certain time, for a period – analogous to a traditional VCR – or alternatively recordings can be programme based.

If an in-progress recording is interrupted and automatically resumed, for example due to a temporary power failure, all subclauses of the recording shall be represented by a single Recording object.

Values of properties in the Recording object shall be obtained from metadata about the recorded programme and are typically copied from the Programme used for scheduling a recording by the `record(Programme programme)` method of the `application/oipfRecordingsScheduler` object. See 7.10.5 for more information about the mapping between the properties of a Programme and the BCG metadata. In the event of a conflict between the metadata in the Programme and that in the broadcast channel, the conflict resolution is implementation dependent.

NOTE The property `parentalRatings` formerly defined as part of this class is now redundant following the renaming of the `parentalRating` property in the `ScheduledRecording` class to `parentalRatings`.

7.10.6.2 Properties

NOTE The properties `state` and `isManual` formerly defined in this class are now defined in the `ScheduledRecording` class, and since the `Recording` class inherits from the `ScheduledRecording` class they are still part of the `Recording` class.

readonly string uri
A uri identifying the content item in local storage according to IETF RFC 3986. The format of the URI is outside the scope of this standard except that; - the scheme shall not be one that is included in this document; - the URI shall not include a fragment.
readonly string id
An identifier for this recording. This value shall be unique to this recording and so can be used to compare two recording objects to see if they refer to the same recording. The OITF shall guarantee that recording identifiers are unique in relation to download identifiers and CODASSET identifiers.
Boolean doNotDelete
If true , then this recording should not be automatically deleted by the system.
Integer saveDays
The number of days for which an individual or manual recording should be saved. Recordings older than this value may be deleted. If the value of this property is undefined , the default save duration shall be used.
Integer saveEpisodes
The number of episodes of a series-link that should be saved. Older episodes may be deleted. This is only valid when set on the latest scheduled recording in the series. If the value of this property is undefined , the default value shall be used.

readonly Boolean **blocked**

Flag indicating whether the programme is blocked due to parental control settings or conditional access restrictions.

The **blocked** and **locked** properties work together to provide a tri-state flag describing the status of a programme. This can best be described by the following table:

Description	blocked	locked
No parental control applies.	false	false
Item is above the parental rating threshold (or manually blocked); no PIN has been entered to view it and so the item cannot currently be viewed.	true	true
Item is above the parental rating threshold (or manually blocked); the PIN has been entered and so the item can be viewed.	true	false
Invalid combination – OITFs shall not support this combination.	false	true

readonly Integer **showType**

Flag indicating the type of show. This field shall take one of the following values:

Value	Description
0	The show is live.
1	The show is a first-run show.
2	The show is a rerun.

readonly Boolean **subtitles**

Flag indicating whether subtitles or closed-caption information is available.

readonly StringCollection **subtitleLanguages**

Supported subtitle languages, indicated by their ISO 639-2 language codes as defined in ISO 639-2.

readonly Boolean **isHD**

Flag indicating whether the programme has high-definition video.

readonly Boolean **is3D**

Flag indicating whether the programme has 3D video.

readonly Integer **audioType**

Bitfield indicating the type of audio that is available for the programme. Since more than one type of audio may be available for a given programme, the value of this field shall consist of one or more of the following values ORed together:

Value	Description
1	Mono audio
2	Stereo audio
4	Multi-channel audio

readonly Boolean isMultilingual
Flag indicating whether more than one audio language is available for this recording.

readonly StringCollection audioLanguages
Supported audio languages, indicated by their ISO 639-2 language codes as defined in ISO 639-2

readonly StringCollection genres
A collection of genres that describe this programme.

readonly Integer recordingStartTime
The actual start time of the recording, including any padding, measured in seconds since midnight (GMT) on 1/1/1970. This may not be the same as the scheduled start time of the recorded programme (e.g. due to a recording starting late, or due to start/end padding). For recordings that have not yet started, the value of this field shall be undefined .

readonly Integer recordingDuration
The actual duration of the recording, including any padding, measured in seconds. This may not be the same as the scheduled duration of the recording (e.g. due to a recording finishing early, or due to start/end padding). For recordings that have not yet started, the value of this field shall be undefined .

readonly BookmarkCollection bookmarks
A collection of the bookmarks set in a recording. If no bookmarks are set, the collection shall be empty.

readonly Boolean locked
Flag indicating whether the current state of the parental control system prevents the recording from being viewed (e.g. a correct parental control PIN has not been entered to allow the recording to be viewed).

7.10.7 The RecordingCollection class

This subclause is intentionally left empty.

7.10.8 The PVREvent class

This subclause is intentionally left empty.

7.10.9 The Bookmark class

7.10.9.1 General

The Bookmark class represents a bookmark or chapter mark in a recording or CoD asset. This is not a web bookmark – instead, it is a point from which the viewer may want to resume playback of a piece of content. These may be set implicitly without user intervention (e.g. at the point where a user stops watching a recording, in order to allow them to resume from that point later) or explicitly by the user (e.g. at the start of a favourite scene).

7.10.9.2 Properties

readonly Integer time
The time at which the bookmark is set, in seconds from the start of the content item.

<code>readonly String name</code>
The name of the bookmark.

7.10.10 The BookmarkCollection class

7.10.10.1 General

```
typedef Collection<Bookmark> BookmarkCollection
```

A `BookmarkCollection` is a collection of bookmarks, ordered by time. See Annex I for the definition of the collection template. In addition to the methods and properties defined for generic collections, the `BookmarkCollection` class supports the additional properties and methods defined below.

In principle bookmarks may be stored on in the network however the protocol for communicating bookmarks between the OITF and the network is not defined in the present document.

7.10.10.2 Methods

<code>Bookmark addBookmark(Integer time, String name)</code>		
Description	Add a new bookmark to the collection. If the bookmark cannot be added (e.g. because the value given for time lies outside the length of the recording), this method shall return <code>null</code> .	
Arguments	time	The time at which the bookmark is set, in seconds since the start of the recording.
	name	The name of the bookmark.

<code>void removeBookmark(Bookmark bookmark)</code>		
Description	Remove a bookmark from the collection.	
Arguments	bookmark	The bookmark to be removed.

7.11 Remote Management APIs

7.11.1 General

This subclause defines interfaces to perform remote diagnostics and management of the device.

Browser based remote management shall be supported by OITFs that have indicated `<remote_diagnostics>true</remote_diagnostics>` in their capability profile (as defined in 9.3.13)

7.11.2 The application/oipfRemoteManagement embedded object

7.11.2.1 General

The `application/oipfRemoteManagement` embedded object has the following properties and methods.

Access to the functionality of the `application/oipfRemoteManagement` embedded object shall adhere to the security requirements as defined in Clause 10.

7.11.2.2 Properties

readonly String **vendorName**

The value of this property shall be the same as the value of the `LocalSystem.vendorName` property (see 7.3.4.2).

readonly String **modelName**

The value of this property shall be the same as the value of the `LocalSystem.modelName` property (see 7.3.4.2).

readonly String **softwareVersion**

The value of this property shall be the same as the value of the `LocalSystem.softwareVersion` property (see 7.3.4.2).

readonly String **hardwareVersion**

The value of this property shall be the same as the value of the `LocalSystem.hardwareVersion` property (see 7.3.4.2).

readonly String **familyName**

The value of this property shall be the same as the value of the `LocalSystem.familyName` property (see 7.3.4.2).

IECNORM.COM : Click to view the full PDF of IEC 62766-5-1:2017

function **onSoftwareUpdate**(String updateEvent, Integer seconds, String message,
String version)

The function that is called when the OITF's software update state is changed. The specified function is called with the following arguments:

- **String updateEvent** – The event type that caused the invocation of this function. One of:

Value	Description
SOFTWARE_AVAILABLE	New software for the OITF has been found on a remote management server. The message argument may contain a user-centric message regarding this new software version and the version argument may contain the version number of the software update.
SOFTWARE_DOWNLOADING	New software for the OITF is in the process of being downloaded. The version argument may contain the version number of the software being downloaded. This event type may be signalled at multiple times during the download of the new software, indicating positive progress; in this case, the message argument should contain an indication of the download progress.
SOFTWARE_DOWNLOAD_FAILED	The download of new software has failed. A descriptive reason for the failure may be found in the message argument and the version argument may contain the version number of the software that failed to be downloaded.
SOFTWARE_DOWNLOADED	A new software version of the OITF has been downloaded but has not yet been installed. Applications can now save relevant data that should survive a firmware upgrade. The actual mechanism of the download is out of scope of this standard. The message argument may contain a user-centric message regarding this new software version. The seconds argument has no significance.
FORCED_UPDATE	A new software version will shortly be installed. This event may occur if the users has not agreed to install the software but the system shall have a new software version. The seconds argument gives the time until the OITF will install the new software.

- **Integer seconds** – The time before action takes place. The meaning depends on the event type as described above.
- **String message** – A message that may be used to inform the user about the purpose of this update to the software in order to receive the users consent to perform the actual update. undefined if not used.
- **String version** – The new version number of the software identified for the update, or undefined if not available.

7.11.2.3 Methods

String getParameter (String parameterName)		
Description	Returns the requested parameter.	
Arguments	parameterName	"SAMPLE_PACKET_LOSS": this queries the RTP packet loss since the last call to this function, or the start of the current RTP content item, whichever is more recent. The returned string is of the format "<time in milliseconds since the last sample> <fraction lost> <number of packets lost>". These fields (i.e. <xxx>) are defined as described in 6.4.2 of IETF RFC 3550, and are decimal numbers (encoded as strings). If no content item is playing, an empty string is returned. "SAMPLE_DECODER_ERRORS": this queries the decoder errors since the last call to this function, or the start of the current RTP content item, whichever is more recent. The returned string is of the format "<time in milliseconds since the sample> <total number of frames decoded> <total number of errors>". These fields are decimal numbers (encoded as strings). If no content item is playing an empty string is returned. "CUMULATIVE_PACKET_LOSS": This queries the RTP packet loss since the start of the current RTP content item. The returned string is of the format "<time in milliseconds of this sample within the content> <fraction lost> <number of packets lost>". These fields (i.e. <xxx>) are defined as described in 6.4.2 of IETF RFC 3550, and are decimal numbers (encoded as strings). If no content item is playing an empty string is returned. "CUMULATIVE_DECODER_ERRORS": This queries the decoder errors since the start of the current RTP content item, whichever is more recent. The returned string is of the format "<time in milliseconds of this sample within the content> <total number of frames decoded> <total number of errors>". These fields are decimal numbers (encoded as strings). If no content item is playing an empty string is returned. Values are not case sensitive. Optionally, further vendor specific parameters may be supported. In the case that a parameter is requested that a device does not support, it shall return an empty string.

Integer triggerSoftwareUpdate (String token)																	
Description	Triggers an OITF to start its software update process. The process itself and any user involvement (e.g. to confirm agreement for a software update) is not defined. The method is blocking. The process of updating the software may generate SoftwareUpdate events to indicate progress. The returned integer is a result code that can take the following values: <table border="1"> <thead> <tr> <th>Result message</th><th>Description</th><th>Semantics</th></tr> </thead> <tbody> <tr> <td>0</td><td>Successful</td><td>The request is successful and the device software will be updated.</td></tr> <tr> <td>1</td><td>Unknown error</td><td>triggerSoftwareUpdate() failed because an unspecified error occurred.</td></tr> <tr> <td>2</td><td>Invalid token</td><td>triggerSoftwareUpdate() failed because the token is not valid.</td></tr> <tr> <td>3</td><td>No update available</td><td>triggerSoftwareUpdate() failed because no update exists.</td></tr> </tbody> </table>		Result message	Description	Semantics	0	Successful	The request is successful and the device software will be updated.	1	Unknown error	triggerSoftwareUpdate() failed because an unspecified error occurred.	2	Invalid token	triggerSoftwareUpdate() failed because the token is not valid.	3	No update available	triggerSoftwareUpdate() failed because no update exists.
Result message	Description	Semantics															
0	Successful	The request is successful and the device software will be updated.															
1	Unknown error	triggerSoftwareUpdate() failed because an unspecified error occurred.															
2	Invalid token	triggerSoftwareUpdate() failed because the token is not valid.															
3	No update available	triggerSoftwareUpdate() failed because no update exists.															
Arguments	token	An optional token string used to assist in the software update process. The token may be used to transfer credentials information to control the software update.															

Integer softwareUpdateStatus()													
Description	Returns the current status of any ongoing software update activity. The value returned by this function shall be:												
	<table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>-2</td><td>N^o software update is in progress.</td></tr> <tr> <td>-1</td><td>New software is available to download for the OITF.</td></tr> <tr> <td>0 ... 99</td><td>New software is being downloaded to the OITF and the value gives an approximation of the amount already downloaded.</td></tr> <tr> <td>100</td><td>Indicates that new software has been successfully downloaded to the OITF and is available for installation.</td></tr> <tr> <td>1001 ... 1999</td><td>Indicates that an error occurred during the download of new software to the OITF. This range of values can be used to provide an implementation specific error code.</td></tr> </table>	Value	Description	-2	N ^o software update is in progress.	-1	New software is available to download for the OITF.	0 ... 99	New software is being downloaded to the OITF and the value gives an approximation of the amount already downloaded.	100	Indicates that new software has been successfully downloaded to the OITF and is available for installation.	1001 ... 1999	Indicates that an error occurred during the download of new software to the OITF. This range of values can be used to provide an implementation specific error code.
Value	Description												
-2	N ^o software update is in progress.												
-1	New software is available to download for the OITF.												
0 ... 99	New software is being downloaded to the OITF and the value gives an approximation of the amount already downloaded.												
100	Indicates that new software has been successfully downloaded to the OITF and is available for installation.												
1001 ... 1999	Indicates that an error occurred during the download of new software to the OITF. This range of values can be used to provide an implementation specific error code.												

7.11.2.4 Events

For the intrinsic events listed in the table below, a corresponding DOM event shall be generated in the following manner:

Intrinsic event	Corresponding DOM event	DOM Event properties
onSoftwareUpdate	SoftwareUpdate	Bubbles: No Cancellable: No Context Info: updateEvent, seconds

The DOM events are directly dispatched to the event target, and will not bubble nor capture. Applications should not rely on receiving these events during the bubbling or the capturing phase. Remote UIs that use DOM event handlers shall call the `addEventListener()` method on the `application/oipfsScheduledRecording` object itself. The third parameter of `addEventListener`, i.e. "useCapture", will be ignored.

7.12 Metadata APIs

7.12.1 General

Subclause 7.12 defines the JavaScript APIs used by DAE applications for reading and searching metadata about programmes. This API does not specify whether these query operations are carried out on the OITF or whether they require communication with a server.

The metadata API provides DAE applications with high-level access to metadata about programmes and channels. This document describes the mapping between this API and CoD and programme guide metadata. Mappings between this API and other metadata sources are not specified in this document.

Subclause 7.12 shall apply for OITFs that have indicated `<clientMetadata>` with value "true" and a "type" attribute with value "bcg" or "dvb-si" as defined in 9.3.8 in their capability.

Note that in order to access the metadata of programmes and channels several extensions to the `Programme` and `Channel` classes have been defined when the OITF has indicated support for `<clientMetadata>`. See 7.16.2.4 and 7.13.12.4 for more information.

The functionality as described in 7.12 is subject to the security model of Clause 10 (in particular 10.1.4.7).

7.12.2 The application/oipfSearchManager embedded object

7.12.2.1 General

OITFs shall implement the "application/oipfSearchManager" embedded object. This object provides a mechanism for applications to create and manage metadata searches.

7.12.2.2 Properties

readonly Integer guideDaysAvailable

The number of days for which guide data is available. A value of –1 means that the amount of guide data available is unknown.

function onMetadataUpdate(Integer action, Integer info, Object object)

This function is the DOM 0 event handler for events indicating changes in metadata. This shall be raised under the following circumstances:

When a new version of the metadata is discovered. Note that new versions of metadata can be made available without any of the individual items of metadata changing. It is an application's responsibility to determine what, if anything, has changed.

When the values of the blocked or locked properties on a content item change due to changes in the parental control subsystem (e.g. parental control being enabled or disabled, or a content item being unlocked with a PIN).

The specified function is called with the arguments action, info and object. These arguments are defined as follows:

- Integer action – the type of update that has taken place. This field will take one of the following values:

Value	Description
1	A new version of metadata is available (see 5.2.3.2.3 of IEC 62766-3:2016) and applications should discard all references to Programme objects immediately and re-acquire them.
2	A change to the parental control flags for a content item has occurred (e.g. the user has unlocked the parental control features of the receiver, allowing a blocked item to be played).
3	A flag affecting the filtering criteria of a channel has changed. Applications may listen for events with this action code to update lists of favourite channels, for instance.

- Integer info – extended information about the type of update that has taken place. If the action argument is set to the value 3, the value of this field shall be one or more of the following:

Value	Description
1	The list of blocked channels has changed.
2	A list of favourite channels has changed.
4	The list of hidden channels has changed.

If the action argument is set to the value 2, the value of this field shall be one or more of:

Value	Description
1	The block status of a content item has changed.
2	The lock status of a content item has changed.

This field is treated as a bitfield, so values may be combined to allow multiple reasons to be passed.

- Object object – the affected channel, programme, or CoD asset prior to the change. If more than one is affected, then this argument shall take the value null.

function onMetadataSearch (MetadataSearch search, Integer state)	
This function is the DOM 0 event handler for events relating to metadata searches. The specified function is called with the arguments search and state . These arguments are defined as follows:	
• MetadataSearch search – the affected search • Integer state – the new state of the search	
Value	Description
0	Search has finished. This event shall be generated when a search has completed.
1	This value is not used.
2	This value is not used.
3	The MetadataSearch object has returned to the idle state, either because of a call to SearchResults.abort() or because the parameters for the search have been modified (e.g. the query, constraints or search target).
4	The search cannot be completed due to a lack of resources or any other reason (e.g. insufficient memory is available to cache all of the requested results).

7.12.2.3 Events

For the intrinsic events listed in the table below, a corresponding DOM event shall be generated in the following manner:

Intrinsic event	Corresponding DOM event	DOM Event properties
onMetadataSearch	MetadataSearch	Bubbles: No Cancellable: No Context Info: search, state
onMetadataUpdate	MetadataUpdate	Bubbles: No Cancellable: No Context Info: action, info, object

These DOM events are directly dispatched to the event target, and will not bubble nor capture. Applications should not rely on receiving these events during the bubbling or the capturing phase. Applications that use DOM event handlers shall call the **addEventListener()** method on the **application/oipfSearchManager** object itself. The third parameter of **addEventListener**, i.e. "useCapture", will be ignored.

7.12.2.4 Methods

MetadataSearch createSearch (Integer searchTarget)								
Description	Create a MetadataSearch object that can be used to search the metadata.							
Arguments	searchTarget	The metadata that should be searched.						
		Valid values of the searchTarget parameter are:						
		<table><tr><th>Value</th><th>Description</th></tr><tr><td>1</td><td>Metadata relating to scheduled content shall be searched.</td></tr><tr><td>2</td><td>Metadata relating to content on demand shall be searched.</td></tr></table>	Value	Description	1	Metadata relating to scheduled content shall be searched.	2	Metadata relating to content on demand shall be searched.
		Value	Description					
1	Metadata relating to scheduled content shall be searched.							
2	Metadata relating to content on demand shall be searched.							
These values are treated as a bitfield, allowing searches to be carried out across multiple search targets.								

ChannelConfig getConfig()	
Description	Returns the channel line-up of the tuner in the form of a ChannelConfig object as defined in 7.13.10. This includes the favourite lists. The ChannelConfig object returned from this function shall be identical to the ChannelConfig object returned from the getConfig() method on the video/broadcast object as defined in 7.13.2.4.

7.12.3 The MetadataSearch class

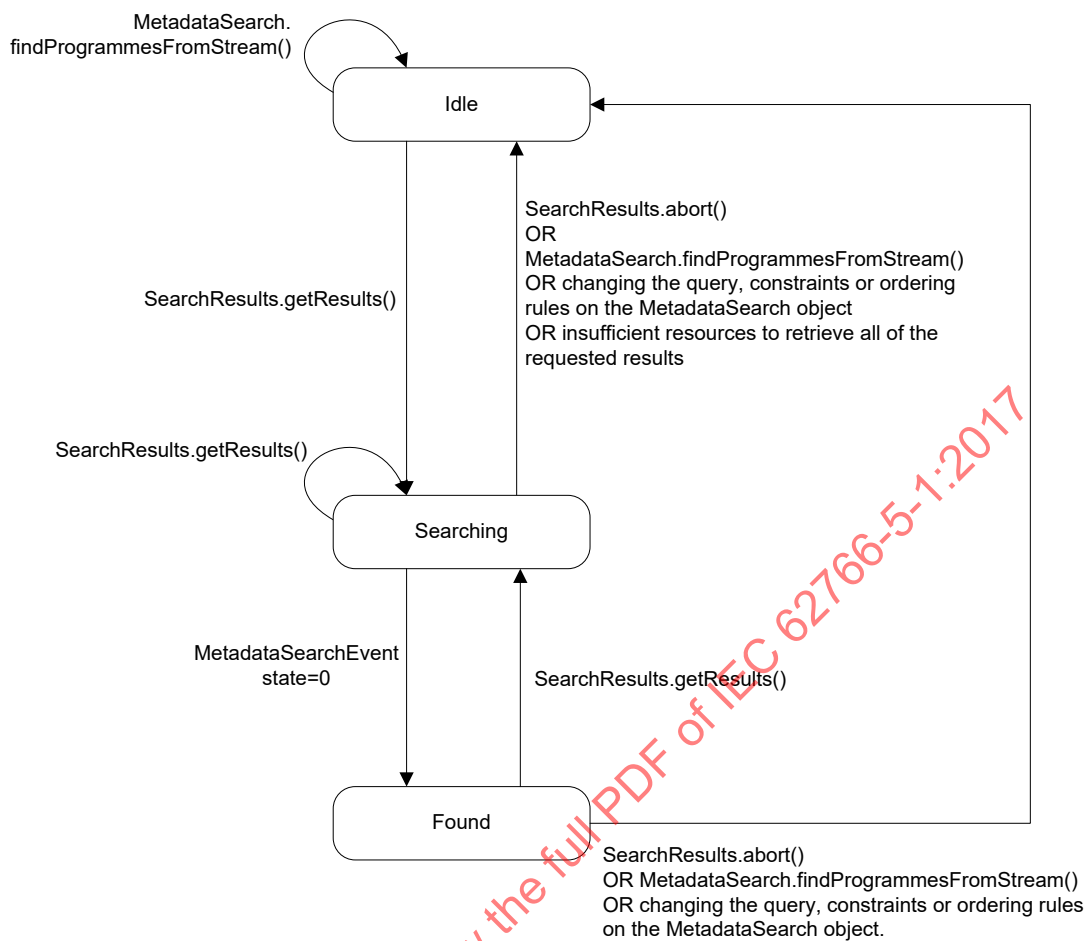
7.12.3.1 General

A **MetadataSearch** object represents a query of the metadata about available programmes. Applications can create **MetadataSearch** objects using the **createSearch()** method on the **application/oipfSearchManager** object. When metadata queries are performed on a remote server, the protocol used is defined in 5.2.3.3 of IEC 62766-3:2016.

Each search consists of three steps:

- Definition of the query. The application creates a **MetadataSearch** object, and either creates its associated **Query** object, or sets a query using the **findProgrammesFromStream()** method, and sets any applicable constraints and result ordering.
- Acquisition of results. The OITF acquires some or all of the items that match the specified query and constraints, and caches the requested subset of the results. This is typically triggered by a call to **getResults()**.
- Retrieval. The application accesses the results via the **SearchResults** class.

The **MetadataSearch** and **SearchResults** classes work together to manage an individual search. For every search, the **MetadataSearch** object and its corresponding **SearchResults** object shall be in one of three states as described in Table 7. Figure 13 shows the transitions between these states.



IEC

Figure 13 – State machine for a metadata search

Table 7 – Metadata search states

State	Description
Idle	The search is idle; no results are available. This is the initial state of the search. In this state, the application can set or modify the query, constraints or ordering rules that are applied to the search. No search results are available in this state – any calls to <code>SearchResults.item()</code> shall return undefined and the values of the <code>length</code> and <code>totalSize</code> properties on the <code>SearchResults</code> object shall return zero. Any search results that have been cached by the terminal shall be discarded when the Idle state is entered. Calling the <code>SearchResults.getResults()</code> method shall cause a state transition to the Searching state.
Searching	Results are being retrieved and are not yet available to applications. If the terminal has not previously cached the full set of search results, the terminal performs the search to gather the requested results. If a new version of the metadata is detected (e.g. due to an EIT update) while the search is in this state, results shall be retrieved from either the new or original version of the metadata but shall not be retrieved from a combination of the two versions. Calls to <code>SearchResults.item()</code> shall return undefined. Any modification of the search parameters (e.g. changing the query or adding/removing constraints, or calling <code>findProgrammesFromStream()</code>) by the application shall stop the current search and cause a transition to the Idle state. The terminal shall dispatch a <code>MetadataSearch</code> event with <code>state=3</code>. When all requested results have been found, the terminal shall dispatch a <code>MetadataSearch</code> event with <code>state=0</code> and a state transition to the Found state shall occur. If the search cannot be completed due to a lack of resources or any other reason, the terminal shall dispatch a <code>MetadataSearch</code> event with <code>state=4</code> and a state transition to the Idle state shall occur. Calls to the <code>SearchResults.getResults()</code> method shall abort the retrieval of search results and attempt to retrieve the newly-requested set of results instead. NOTE Calling <code>getResults()</code> when in the searching state may be used to fetch a group of items starting at a different offset or with a different count.
Found	Search results are available and can be retrieved by applications. The data exposed via the <code>SearchResults.item()</code> method is static and never changes as a result of any updates to the underlying metadata database until <code>SearchResults.getResults()</code> is next called. If a new version of the metadata is detected (e.g. due to an EIT update), a <code>MetadataUpdate</code> event is dispatched with <code>action=1</code>. Subsequent calls to <code>SearchResult.getResults()</code> shall return results based on the updated metadata. Calls to <code>SearchResults.getResults()</code> shall cause a state transition to the Searching state. Any modification of the search parameters (e.g. changing the query or adding/removing constraints, or calling <code>findProgrammesFromStream()</code>) by the application shall cause the current set of results to be discarded and shall cause a transition to the Idle state. The terminal shall dispatch a <code>MetadataSearch</code> event with <code>state=3</code>.

The `findProgrammesFromStream()` method acts as a shortcut for setting a query and a set of constraints on the `MetadataSearch` object. Regardless of whether the query and constraints are set explicitly by the application or via `findProgrammesFromStream()`, results are retrieved using the `getResults()` method.

Changes to the search parameters (e.g. changing the query or adding/removing constraints or modifying the search target, or calling `findProgrammesFromStream()`) shall be applied when the `getResults()` method on the corresponding `SearchResults` object is called. Due to the nature of metadata queries, searches are asynchronous and events are used to notify the application that results are available. `MetadataSearch` events shall be targeted at the application/`oipfSearchManager` object.

The present document is intentionally silent about the implementation of the search mechanism and the algorithm for retrieving and caching search results except where described in Table 7 above. When performing a search, the receiver may gather all search results and cache them (or cache a set of pointers into the full database), or gather only the subset of search results determined by the `getResults()` parameters, or take an alternative approach not described here.

7.12.3.2 Properties

readonly Integer searchTarget	
The target(s) of the search. Valid values are:	
Value	Description
1	Metadata relating to scheduled content shall be searched.
2	Metadata relating to on-demand content shall be searched.
These values shall be treated as a bitfield, allowing searches to be carried out across multiple search targets.	

readonly SearchResults result
The subset of search results that has been requested by the application.

7.12.3.3 Methods

void setQuery(Query query)		
Description	Set the query terms to be used for this search, discarding any previously-set query terms. Setting the search parameters using this method will implicitly remove any existing constraints, ordering or queries created by prior calls to methods on this object.	
Arguments	query	The query terms to be used

void addRatingConstraint(ParentalRatingScheme scheme, Integer threshold)		
Description	Constrain the search to only include results whose parental rating value is below the specified threshold.	
Arguments	scheme	The parental rating scheme upon which the constraint shall be based. If the value of this argument is <code>null</code> , any existing parental rating constraints shall be cleared.
	threshold	The threshold above which results shall not be returned. If the value of this argument is <code>null</code> , any existing constraint for the specified parental rating scheme shall be cleared.

void addCurrentRatingConstraint()	
Description	Constrain the search to only include results whose parental rating value is below the threshold currently set by the user.

void addChannelConstraint(ChannelList channels)		
Description	Constrain the search to only include results from the specified channels. If a channel constraint has already been set, subsequent calls to <code>addChannelConstraint()</code> shall add the specified channels to the list of channels from which results should be returned. For CoD searches, adding a channel constraint shall have no effect.	
Arguments	channels	The channels from which results shall be returned. If the value of this argument is <code>null</code> , any existing channel constraint shall be removed.

void addChannelConstraint(Channel channel)		
Description	Constrain the search to only include results from the specified channel. If a channel constraint has already been set, subsequent calls to addChannelConstraint() shall add the specified channel to the list of channels from which results should be returned. For CoD searches, adding a channel constraint shall have no effect.	
Arguments	channel	The channel from which results shall be returned. If the value of this argument is null , any existing channel constraint shall be removed.

void orderBy(String field, Boolean ascending)		
Description	Set the order in which results should be returned in future. Any existing search results shall not be re-ordered. Subsequent calls to orderBy() will apply further levels of ordering within the order defined by previous calls. For example: <pre>orderBy("ServiceName", true); orderBy("PublishedStart", true);</pre> will cause results to be ordered by service name and then by start time for results with the same channel number.	
Arguments	field	The name of the field by which results should be sorted. A value of null indicates that any currently-set order shall be cleared and the default sort order should be used.
	ascending	Flag indicating whether the results should be returned in ascending or descending order.

Query createQuery(String field, Integer comparison, String value)																		
Description	Create a metadata query for a specific value in a specific field within the metadata. Simple queries may be combined to create more complex queries. Applications shall follow the JavaScript type conversion rules to convert non-string values into their string representation, if necessary.																	
Arguments	field	The name of the field to compare. Fields are identified using the format <classname>.<propertyname> where classname shall be one of "Programme", "CODAsset", "CODService" or "CODFolder" and <propertyname> shall be a valid property name on the corresponding class.																
	comparison	<table><tr><th>Value</th><th>Description</th></tr><tr><td>0</td><td>True if the specified value is equal to the value of the specified field.</td></tr><tr><td>1</td><td>True if the specified value is not equal to the value of the specified field.</td></tr><tr><td>2</td><td>True if the value of the specified field is greater than the specified value.</td></tr><tr><td>3</td><td>True if the value of the specified field is greater than or equal to the specified value.</td></tr><tr><td>4</td><td>True if the value of the specified field is less than the specified value.</td></tr><tr><td>5</td><td>True if the value of the specified field is less than or equal to the specified value.</td></tr><tr><td>6</td><td>True if the string value of the specified field contains the specified value. This operation shall be case insensitive, and shall match parts of a word as well as whole words (e.g. a value of "term" will match a field value of "Terminator").</td></tr></table>	Value	Description	0	True if the specified value is equal to the value of the specified field.	1	True if the specified value is not equal to the value of the specified field.	2	True if the value of the specified field is greater than the specified value.	3	True if the value of the specified field is greater than or equal to the specified value.	4	True if the value of the specified field is less than the specified value.	5	True if the value of the specified field is less than or equal to the specified value.	6	True if the string value of the specified field contains the specified value. This operation shall be case insensitive, and shall match parts of a word as well as whole words (e.g. a value of "term" will match a field value of "Terminator").
	Value	Description																
0	True if the specified value is equal to the value of the specified field.																	
1	True if the specified value is not equal to the value of the specified field.																	
2	True if the value of the specified field is greater than the specified value.																	
3	True if the value of the specified field is greater than or equal to the specified value.																	
4	True if the value of the specified field is less than the specified value.																	
5	True if the value of the specified field is less than or equal to the specified value.																	
6	True if the string value of the specified field contains the specified value. This operation shall be case insensitive, and shall match parts of a word as well as whole words (e.g. a value of "term" will match a field value of "Terminator").																	
	value	The value to check. Applications shall follow the JavaScript type conversion rules to convert non-string values into their string representation, if necessary.																

void findProgrammesFromStream(Channel channel, Integer startTime, Integer count)		
Description	Set a query and constraints for retrieving metadata for programmes from a given channel and given start time from metadata contained in the stream as defined in 5.2.4 of IEC 62766-3:2016. Setting the search parameters using this method will implicitly remove any existing constraints, ordering or queries created by prior calls to methods on this object. This method does not cause the search to be performed; applications shall call getResults() to retrieve the results. Applications shall be notified of the progress of the search via MetadataSearch events as described in 7.12.3.	
Arguments	channel	The channel for which programme information should be found.
	startTime	The start of the time period for which results should be returned measured in seconds since midnight (GMT) on 1/1/1970. The start time is inclusive; any programmes starting at the start time, or which are showing at the start time, will be included in the search results. If null , the search will start from the current time.
	count	Optional argument giving the maximum number of programmes for which information should be fetched. This places an upper bound on the number of results that will be present in the result set – for instance, specifying a value of 2 for this argument will result in at most two results being returned by calls to getResults() even if a call to getResults() requests more results. If this argument is not specified, no restrictions are imposed on the number of results which may be returned by calls to getResults().

7.12.4 The Query class

7.12.4.1 General

The Query class represents a metadata query that the user wants to carry out. This may be a simple search, or a complex search involving Boolean logic. Queries are immutable; an operation on a query shall return a new query object, allowing applications to continue referring to the original query.

The examples below show how more complex queries can be constructed:

```
Query qa = mySearch.createQuery("Title", 6, "Terminator");
Query qb = mySearch.createQuery("SpokenLanguage", 0, "fr-CA");
Query qc = qb.and(qa.not());
```

7.12.4.2 Properties

This subclause is intentionally left empty.

7.12.4.3 Methods

Query and(Query query)		
Description	Create a query based on the logical AND of the predicates represented by the query currently being operated on and the argument query .	
Arguments	query	The second predicate for the AND operation.

Query or(Query query)		
Description	Create a query based on the logical OR of the predicates represented by the query currently being operated on and the argument query .	
Arguments	query	The second predicate for the OR operation.

Query not()	
Description	Create a query that is the logical negation of the predicates represented by the query currently being operated on.

7.12.5 The SearchResults class

7.12.5.1 General

The `SearchResults` class represents the results of a metadata search. Since the result set may contain a large number of items, applications request a 'window' on to the result set, similar to the functionality provided by the `OFFSET` and `LIMIT` clauses in SQL.

Applications may request the contents of the result in groups of an arbitrary size, based on an offset from the beginning of the result set. The data shall be fetched from the appropriate source, and the application shall be notified when the data is available.

The set of results shall only be valid if a call to `getResults()` has been made and a `MetadataSearch` event notifying the application that results are available has been dispatched. If this event has not been dispatched, the set of results shall be empty (i.e. the value of the `totalSize` property shall be 0 and calls to `item()` shall return undefined).

In addition to the properties and methods defined below, a `SearchResults` object shall support the array notation to access the results in this collection.

7.12.5.2 Properties

readonly Integer length
The number of items in the current window within the overall result set. The value of this property shall be zero until <code>getResults()</code> has been called and a <code>MetadataSearch</code> event notifying the application that results are available has been dispatched. If the current window onto the result set is in fact the whole result set then length will be the same as totalSize . Otherwise length will be less than totalSize .

readonly Integer offset
The current offset into the total result set.

readonly Integer totalSize
The total number of items in the result set. The value of this property shall be zero until <code>getResults()</code> has been called and a <code>MetadataSearch</code> event notifying the application that results are available has been dispatched.

7.12.5.3 Methods

Object item(Integer index)		
Description	Return the item at position index in the collection of currently available results, or undefined if no item is present at that position. This function shall only return objects that are instances of <code>Programme</code> , <code>CODAsset</code> , <code>CODFolder</code> , or <code>CODService</code> .	
Arguments	index	The index into the result set.

Boolean getResults(Integer offset, Integer count)		
Description	Perform the search and retrieve the specified subset of the items that match the query. Results shall be returned asynchronously. A <code>MetadataSearch</code> event with <code>state=0</code> shall be dispatched when results are available. This method shall always return false .	
Arguments	offset	The number of items at the start of the result set to be skipped before data is retrieved.
	count	The number of results to retrieve.

void abort()	
Description	Abort any outstanding request for results and remove any query, constraints or ordering rules set on the MetadataSearch object that is associated with this SearchResults object. Regardless of whether or not there is an outstanding request for results, items currently in the collection shall be removed (i.e. the value of the length property shall be 0 and any calls to item() shall return undefined). All cached search results shall be discarded.

7.12.6 The MetadataSearchEvent class

This subclause is intentionally left empty.

7.12.7 The MetadataUpdateEvent class

This subclause is intentionally left empty.

7.13 Scheduled content and hybrid tuner APIs

7.13.1 General

Subclause 7.13 shall apply to OITFs that have indicated support for tuner control (i.e. `<video_broadcast>true</video_broadcast>` as defined in 9.3.2) in their capability. It describes the video/broadcast embedded object needed to support display and control by a DAE application of scheduled content received over local tuner functionality available to an OITF, including the conveyance of the channel list to the server. The term "tuner" is used here to identify a piece of functionality to enable switching between different types of scheduled content services that are identified through logical channels. This includes IP broadcast channels, as well as traditional broadcast channels received over a hybrid tuner.

7.13.2 The video/broadcast embedded object

7.13.2.1 General

The OITF shall support the video/broadcast embedded object with the following properties and methods, which shall adhere to the tuner related security requirements in 10.1.4.2. The MIME type of this object shall be "video/broadcast".

7.13.2.2 State diagram for video/broadcast objects

The state diagram in Figure 14 shows the states that a video/broadcast object may be in. Dashed lines indicate automatic transitions between states. The video/broadcast object shall be in the `unrealized` state when it is instantiated.

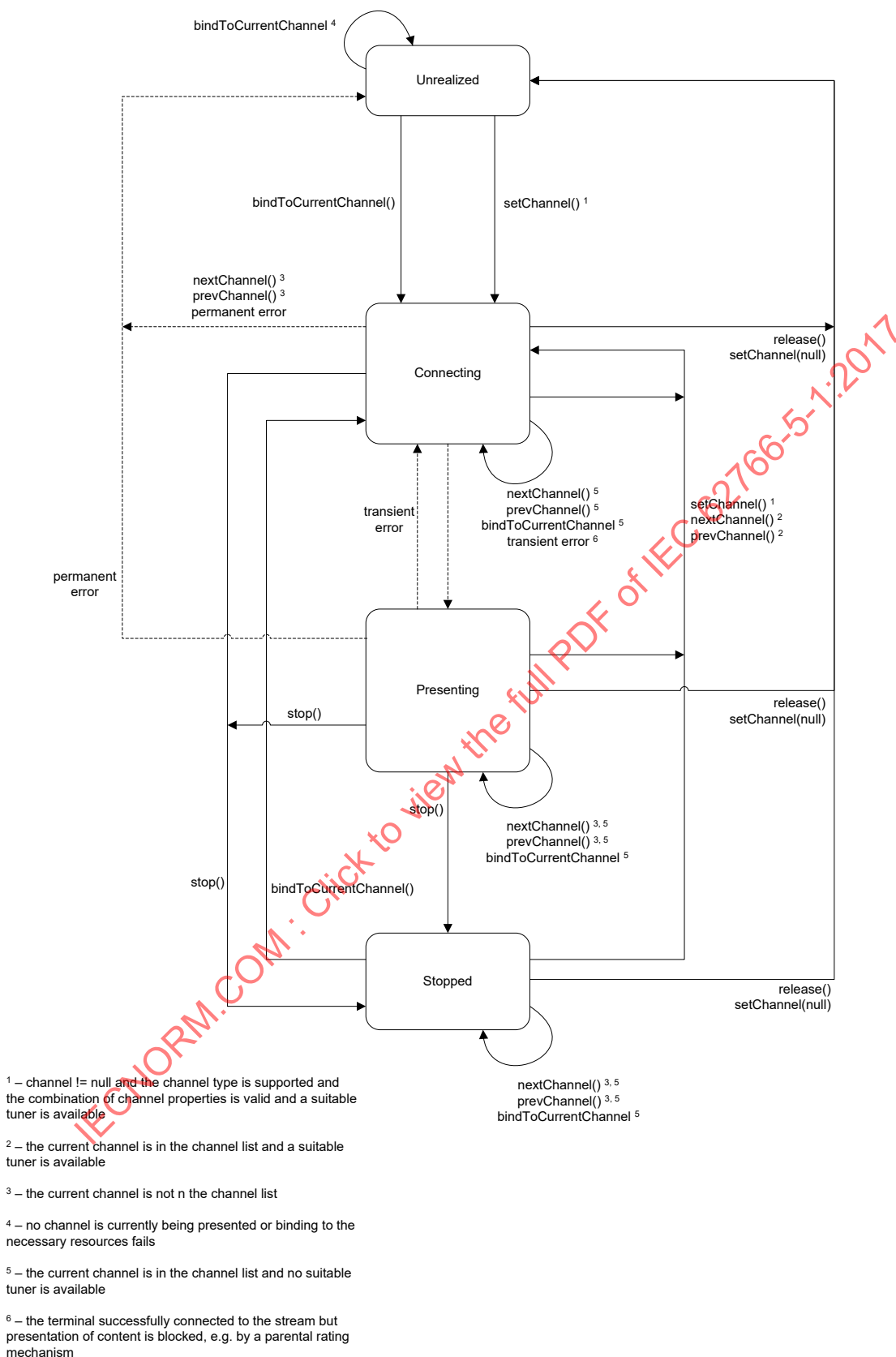


Figure 14 – State diagram for embedded video/broadcast objects

Transient errors are defined as ones that the OITF will automatically recover from without intervention by an application. Transient errors persist until either the condition which caused them is corrected or it is determined that it cannot be connected and the error becomes

permanent. Permanent errors are defined as ones that the OITF will not automatically attempt to recover from.

Terminals shall perform the state changes in Table 8 under the conditions described and generate the listed event(s). Terminals shall not change state in circumstances other than defined in this subclause.

Table 8 – State transitions for the video/broadcast embedded object

Old State	Trigger	New State	State Transition Events	Description
All states	setChannel(channel) where channel != null and the channel type is supported and the combination of channel properties is valid and a suitable tuner is available	Connecting	PlayStateChange	The terminal attempts to connect to the requested channel. The currentChannel object reflects the channel being changed to.
All states	setChannel(channel) where channel != null but either the channel type is not supported or the combination of channel properties is invalid or a suitable tuner is not available	No change	ChannelChangeError	The terminal remains in the same state.
Connecting or Presenting or Stopped	nextChannel(), prevChannel() where the video/broadcast object currentChannel is in the channel list and a suitable tuner is available	Connecting	PlayStateChange	The terminal attempts to connect to the requested channel. The currentChannel object reflects the channel being changed to.
Connecting	nextChannel(), prevChannel() where the video/broadcast object currentChannel is not in the channel list	Unrealized	ChannelChangeError PlayStateChange	
Presenting or Stopped	nextChannel(), prevChannel(), where the video/broadcast object currentChannel is not in the channel list	No change	ChannelChangeError	The terminal remains in the same state.
Connecting or Presenting or Stopped	nextChannel(), prevChannel() where the video/broadcast object currentChannel is in the channel list but no suitable tuner is available	No change	ChannelChangeError	The terminal remains in the same state.
Unrealized	bindToCurrentChannel() when at least one channel is currently being presented by the OITF and binding to the necessary resources does not fail	Presenting	PlayStateChange	The terminal binds the video/broadcast object to the current channel being natively presented. The currentChannel object reflects the channel being presented.
Unrealized	bindToCurrentChannel() when there is no channel currently being presented or binding to the necessary resources to play the channel through the video/broadcast object fails	Unrealized	PlayStateChange	The terminal continues to present the current channel, if any.

Old State	Trigger	New State	State Transition Events	Description
Connecting	The terminal successfully connected to the broadcast or IP multicast stream and presented its contents.	Presenting	ChannelChangeSucceeded PlayStateChange	This transition occurs automatically when media presentation starts.
Connecting	The terminal successfully connected to the broadcast or IP multicast stream but presentation of content is blocked, e.g. by a parental rating mechanism or content protection mechanism	Connecting	ChannelChangeSucceeded PlayStateChange	This is conceptually equivalent to a successful channel change where a transient error immediately pre-empts media presentation without the video/broadcast object entering the presenting state.
Connecting	Recovery from a transient error, including – presentation of content no longer being blocked by a content protection mechanism (e.g. the start of a free preview period or a channel that changes from being encrypted to being in the clear during the day) – the end-user entering a PIN code or other equivalent authorization to enable access to content protected by parental access control – resumption of delivery of media data	Presenting	PlayStateChange	If a video/broadcast object was forced from the presenting state to the connecting state due to a transient error and that error condition clears while the video/broadcast object remains in the connecting state then the video/broadcast object shall automatically transition back to the presenting state.
Connecting or Presenting or Stopped	release() or setChannel(null)	Unrealized	PlayStateChange	The control is returned to the terminal. The currentChannel object is set to null. If an application has modified the set of components being presented (e.g. changing the audio or subtitle stream being presented) then the same set of components will continue to be presented.

Old State	Trigger	New State	State Transition Events	Description
Connecting	Permanent error including – failure to change to a new channel (e.g. the channel cannot be found or none of the media components can be decoded or insufficient resources are available to present the channel) – exhaustion of all possibilities for an end-user to authorize access to content protected by a parental access control mechanism (e.g. timeout on a PIN entry dialogue) – delivery of media data was interrupted and has not resumed after an implementation-dependent timeout	Unrealized	ChannelChangeError PlayStateChange	The terminal encountered a permanent error
Connecting or Presenting	stop()	Stopped	PlayStateChange	
Presenting	Transient error including – presentation of content being blocked by a content protection mechanism, – presentation of content being blocked by a parental rating mechanism, – interruption of delivery of media data (either via IP or hybrid) if either; a) the media data is delivered over a connection and the connection remains intact or b) the media data is delivered via a connectionless mechanism	Connecting	PlayStateChange	The terminal encountered a transient error. During media presentation, transient errors (e.g. transient errors in the bitstream, temporary loss of signal or temporary halting of media decoding due to parental control issues) may cause the object to transition from the presenting state to the connecting state. Temporary loss of resources due to presentation being interrupted by playback of audio from memory may cause the object to transition from the presenting state to the connecting state.
Presenting or Stopped	Permanent error including; – interruption of delivery of media data where the media data is delivered over a connection and the connection terminates	Unrealized	PlayStateChange	The terminal encountered a permanent error.
Stopped	bindToCurrentChannel()	Connecting	PlayStateChange	Video and audio presentation is enabled.

Old State	Trigger	New State	State Transition Events	Description
All states	Destroy video/broadcast	N/A		When a video/broadcast object is destroyed (e.g. by the video/broadcast object being garbage collected) control of broadcast video shall be returned to the terminal. If an application has modified the set of components being presented (e.g. changing the audio or subtitle stream being presented) then the same set of components will continue to be presented. When a video/broadcast object is destroyed due to a page transition within an application, terminals may delay this operation until the new page is fully loaded in order to avoid display glitches if a video/broadcast object is also present in the new page. Presentation of broadcast video or audio shall not be interrupted in either case.

If the channel currently being presented is requested to be changed due to an action outside the application (for example, the user pressing the CH+ key on the remote) then any video/broadcast object presenting that channel (e.g. as the result of a call to `bindToCurrentChannel()`) shall perform the same state transitions and dispatch the same events as if the channel change operation was initiated by the application using the `setChannel()` method.

If the value of the `allocationMethod` property is `DYNAMIC_ALLOCATION`, the following apply.

- Scarce resources such as media decoders shall be claimed while in the connecting state.
- Resources shall be released when the video/broadcast object transitions to the unrealized state.
- Video and audio decoding resources shall be released when the video/broadcast object transitions to the stopped state.
- Transitioning from the presenting to the connecting state should not cause scarce resources to be released.

Applications can use the `playState` property of the video/broadcast object to read its current state.

When a video/broadcast object stops being rendered as defined in 10.1 of the HTML5 specification as referenced by IEC 62766-5-2, an OITF may release scarce resources allocated for that object. Vice versa, a video/broadcast object which is not visible but it's still being rendered shall still be decoding video if it is in the presenting state and any audio associated with the currently presented channel will still be audible. State transitions caused by calls to methods on the video/broadcast object, or due to permanent or transient errors, will occur as shown above regardless of the visibility of the object

NOTE As implied by the text above, rendering state and visibility are not equivalent. This implies, just to give two examples, that `display:none` will affect the object state while `visibility:hidden` will not.

7.13.2.3 Properties

Integer width
The width of the area used for rendering the video object. This property is only writable if property fullScreen has value false . Changing the width property corresponds to changing the width property through the HTMLObjectElement interface, and shall have the same effect as changing the width through the DOM Level 2 Style interfaces (i.e. CSS2Properties interface style.width), at least for values specified in pixels.
Integer height
The height of the area used for rendering the video object. This property is only writable if property fullScreen has value false . Changing the height property corresponds to changing the height property through the HTMLObjectElement interface, and shall have the same effect as changing the height through the DOM Level 2 Style interfaces (i.e. CSS2Properties interface style.height), at least for values specified in pixels.
readonly Boolean fullScreen
Returns true if this video object is in full-screen mode, false otherwise. The default value is false .
String data
Setting the value of the data property shall have no effect on the video/broadcast object. If this property is read, the value returned shall always be the empty string.

IECNORM.COM : Click to view the full PDF of IEC 62766-5-1:2017

function **onChannelChangeError**(Channel channel, Number errorState)

The function that is called when a request to switch a tuner to another channel resulted in an error preventing the broadcasted content from being rendered. The specified function is called with the arguments **channel** and **errorState**. This function may be called either in response to a channel change initiated by the application, or a channel change initiated by the OITF (see 7.13.2.2). These arguments are defined as follows:

- **Channel channel** – the Channel object to which a channel switch was requested, but for which the error occurred. This object shall have the same properties as the channel that was requested, except that for channels of type ID_DVB_* the values for the **onid** and **tsid** properties shall be extracted from the transport stream when one was found (e.g. when **errorState** is 12).

Number errorState – error code detailing the type of error:

Value	Description
0	Channel not supported by tuner.
1	Cannot tune to given transport stream (e.g. no signal)
2	Tuner locked by other object.
3	Parental lock on channel.
4	Encrypted channel, key/module missing.
5	Unknown channel (e.g. can't resolve DVB or ISDB triplet).
6	Channel switch interrupted (e.g. because another channel switch was activated before the previous one completed).
7	Channel cannot be changed, because it is currently being recorded.
8	Cannot resolve URI of referenced IP channel.
9	Insufficient bandwidth.
10	Channel cannot be changed by nextChannel() / prevChannel() methods either because the OITF does not maintain a favourites or channel list or because the video/broadcast object is in the Unrealized state.
11	Insufficient resources are available to present the given channel (e.g. a lack of available codec resources).
12	Specified channel not found in transport stream.
100	Unidentified error.

readonly Integer **playState**

The current play state of the **video/broadcast** object. Valid values are:

Value	Description
0	Unrealized; the application has not made a request to start presenting a channel or has stopped presenting a channel and released any resources. The content of the video/broadcast object should be transparent but if not shall be an opaque black rectangle. Control of media presentation is under the control of the OITF, as defined in Clause F.2.
1	Connecting; the terminal is connecting to the media source in order to begin playback. Objects in this state may be buffering data in order to start playback. Control of media presentation is under the control of the application, as defined in Clause F.2. The content of the video/broadcast object is implementation dependent.
2	Presenting; the media is currently being presented to the user. The object is in this state regardless of whether the media is playing at normal speed, paused, or playing in a trick mode (e.g. at a speed other than normal speed). Control of media presentation is under the control of the application, as defined in Clause F.2. The video/broadcast object contains the video being presented.
3	Stopped; the terminal is not presenting media, either inside the video/broadcast object or in the logical video plane. The logical video plane is disabled. The content of the video/broadcast object shall be an opaque black rectangle. Control of media presentation is under the control of the application, as defined in Clause F.2

See 7.13.2.2 for a description of the state model for a **video/broadcast** object.

Note that the implementations where the content of the **video/broadcast** object is transparent in the unrealized state will give a better user experience than ones where it is black. This happens for an application with video in the background between when it includes a **video/broadcast** object in the page and when a call to **bindToCurrentChannel()** completes. Applications which do not need to call **bindToCurrentChannel()** should not do so. The current channel can be obtained from the **currentChannel** property on the **ApplicationPrivateData** object which is the same as that on the **video/broadcast** object under most normal conditions.

function **onPlayStateChange**(Number state, Number error)

The function that is called when the play state of the **video/broadcast** object changes. This function may be called either in response to an action initiated by the application, an action initiated by the OITF or an error (see 7.13.2.2).

The specified function is called with the arguments **state** and **error**. These arguments are defined as follows:

- **Number state** – the new state of the video/broadcast object. Valid values are given in the definition of the **playState** property above.
- **Number error** – if the state has changed due to an error, this field contains an error code detailing the type of error. See the definition of **onChannelChangeError** above for valid values. If no error has occurred, this argument shall take the value undefined.

function **onChannelChangeSucceeded**(channel channel)

The function that is called when a request to switch a tuner to another channel has successfully completed. This function may be called either in response to a channel change initiated by the application, or a channel change initiated by the OITF (see 7.13.2.2). The specified function is called with argument **channel**, which is defined as follows:

- **channel channel** – the channel to which the tuner switched. This object shall have the same properties with the same values as the **currentChannel** object (see 7.13.8).

function **onFullScreenChange**()

The function that is called when the value of **fullScreen** changes.

function **onfocus**()

The function that is called when the video object gains focus.

function onblur()
The function that is called when the video object loses focus.

readonly StringCollection playerCapabilities
The list of media formats that are supported by the object. Each item shall contain a format label according to IEC 62766-2-1.
If scarce resources are not claimed by the object, the value of this property shall be null .

readonly Integer allocationMethod
Returns the resource allocation method currently in use by the object. Valid values as defined in 7.14.14.2 are:
• STATIC_ALLOCATION • DYNAMIC_ALLOCATION

7.13.2.4 Methods

ChannelConfig getConfig()	
Description	Returns the channel line-up of the tuner in the form of a ChannelConfig object as defined in 7.13.10. The method shall return the value null if the channel list is not (partially) managed by the OITF (i.e., if the channel list information is managed entirely in the network).

Channel bindToCurrentChannel()	
Description	If the video/broadcast object is in the unrealized state and video from exactly one channel is currently being presented by the OITF then this binds the video/broadcast object to that video. If the video/broadcast object is in the stopped state, then this restarts presentation of video and audio from the current channel under the control of the video/broadcast object. If video from more than one channel is currently being presented by the OITF, then this binds the video/broadcast object to the channel whose audio is being presented. If there is no channel currently being presented, or binding to the necessary resources to play the channel through the video/broadcast object fails for whichever reason, the OITF shall dispatch an event to the onPlayStateChange listener(s) whereby the state parameter is given value 0 ("unrealized") and the error parameter is given the appropriate error code. Calling this method from any other states than the unrealized or stopped states shall have no effect. See 7.13.2.2 for more information of its usage. Returning a Channel object from this method does not guarantee that video or audio from that channel is being presented. Applications should listen for the video/broadcast object entering state 2 ("presenting") in order to determine when audio or video is being presented.

Channel createChannelObject (Integer idType, String dsd, Integer sid)		
Description	Creates a Channel object of the specified idType. This method is typically used to create a Channel object of type ID_DVB_SI_DIRECT. The Channel object can subsequently be used by the setChannel() method to switch a tuner to this channel, which may or may not be part of the channel list in the OITF. The resulting Channel object represents a locally defined channel which, if not already present there, does not get added to the channel list accessed through the ChannelConfig class (see 7.13.10). Valid value for idType include: ID_DVB_SI_DIRECT. For other values this behaviour is not specified. If the channel of the given type cannot be created or the delivery system descriptor is not valid, the method shall return null. If the channel of the given type can be created and the delivery system descriptor is valid, the method shall return a Channel object whereby at a minimum the properties with the same names (i.e. idType, dsd and sid) are given the same value as argument idType, dsd and sid of the createChannelObject method.	
Arguments	idType	The type of channel, as indicated by one of the ID_* constants defined in 7.13.12.2. Valid values for idType include: ID_DVB_SI_DIRECT . For other values this behaviour is not specified.
	dsd	The delivery system descriptor (tuning parameters) represented as a string whose characters shall be restricted to the ISO Latin-1 character set. Each character in the dsd represents a byte of a delivery system descriptor as defined by DVB-SI ETSI EN 300 468, 6.2.13, such that a byte at position "i" in the delivery system descriptor is equal the Latin-1 character code of the character at position "i" in the dsd .
	sid	The service ID, which shall be within the range of 1 to 65535.

Channel createChannelObject (Integer idType, Integer onid, Integer tsid, Integer sid, Integer sourceID, String ipBroadcastID)		
Description	Creates a Channel object of the specified idType. The Channel object can subsequently be used by the setChannel() method to switch a tuner to this channel, which may or may not be part of the channel list in the OITF. The resulting Channel object represents a locally defined channel which, if not already present there, does not get added to the channel list accessed through the ChannelConfig class (see 7.13.10). If the channel of the given idType cannot be created or the given (combination of) arguments are not considered valid or complete, the method shall return null. If the channel of the given type can be created and arguments are considered valid and complete, then either: • If the channel is in the channel list then a new object of the same type and with properties with the same values shall be returned as would be returned by calling getChannelwithTriplet() with the same parameters as this method. • Otherwise, the method shall return a Channel object whereby at a minimum the properties with the same names are given the same value as the given arguments of the createChannelObject() method. The values specified for the remaining properties of the Channel object are set to undefined.	
Arguments	idType	The type of channel, as indicated by one of the ID_* constants defined in 7.13.12.2.
	onid	The original network ID. Optional argument that shall be specified when the idType specifies a channel of type ID_DVB_*, ID_IPTV_URI, or ID_ISDB_* and shall otherwise be ignored by the OITF.
	tsid	The transport stream ID. Optional argument that may be specified when the idType specifies a channel of type ID_DVB_*, ID_IPTV_URI, or ID_ISDB_* and shall otherwise be ignored by the OITF.
	sid	The service ID. Optional argument that shall be specified when the idType specifies a channel of type ID_DVB_*, ID_IPTV_URI, or ID_ISDB_* and shall otherwise be ignored by the OITF.
	sourceID	The source_ID. Optional argument that shall be specified when the idType specifies a channel of type ID_ATSC_T and shall otherwise be ignored by the OITF.
	ipBroadcastID	The DVB textual service identifier of the IP broadcast service, specified in the format "ServiceName.DomainName" when idType specifies a channel of type ID_IPTV_SDS, or the URI of the IP broadcast service when idType specifies a channel of type ID_IPTV_URI. Optional argument that shall be specified when the idType specifies a channel of type ID_IPTV_SDS or ID_IPTV_URI and shall otherwise be ignored by the OITF.

void	setChannel(Channel channel, Boolean trickplay, String contentAccessDescriptorURL)
Description	Requests the OITF to switch a (logical or physical) tuner to the channel specified by channel and render the received broadcast content in the area of the browser allocated for the video/broadcast object. If the channel specifies an idType attribute value which is not supported by the OITF or a combination of properties that does not identify a valid channel, the request to switch channel shall fail and the OITF shall trigger the function specified by the onChannelChangeError property, specifying the value 0 ("Channel not supported by tuner") for the errorState, and dispatch the corresponding DOM event (see below). If the channel specifies an idType attribute value supported by the OITF, and the combination of properties defines a valid channel, the OITF shall relay the channel switch request to a local physical tuner that is currently not in use by another video/broadcast object and that can tune to the specified channel. If no tuner satisfying these requirements is available (i.e. all physical tuners that could receive the specified channel are in use), the request shall fail and OITF shall trigger the function specified by the onChannelChangeError property, specifying the value '2' ("tuner locked by other object") for the errorState and dispatch the corresponding DOM event (see below). If multiple tuners satisfying these requirements are available, the OITF selects one. If the channel specifies an IP broadcast channel, and the OITF supports idType ID_IPTV_SDS or ID_IPTV_URI, the OITF shall relay the channel switch request to a logical 'tuner' that can resolve the URI of the referenced IP broadcast channel. If no logical tuner can resolve the URI of the referenced IP broadcast channel, the request shall fail and the OITF should trigger the function specified by the onChannelChangeError property, specifying the value 8 ("cannot resolve URI of referenced IP channel") for the errorState, and dispatch the corresponding DOM event. The optional attribute contentAccessDescriptorURL allows for the inclusion of a Content Access Streaming Descriptor (the format of which is defined in Clause C.2) to provide additional information for dealing with IPTV broadcasts that are (partially) DRM-protected. The descriptor may for example include Marlin action tokens or a previewLicense. The attribute shall be undefined or null if it is not applicable. If the attribute contentAccessDescriptorURL is present, the trickplay attribute shall take a value of either true or false. If the Transport Stream cannot be found, either via the DSD or the (ONID,TSID) pair, then a call to onChannelChangeError with errorstate=5 ("unknown channel") shall be triggered, and the corresponding DOM event dispatched. If the OITF succeeds in tuning to a valid transport stream but this transport stream does not contain the requested service in the PAT, the OITF shall remain tuned to that location and shall trigger a call to onChannelChangeError with errorstate=12 ("specified channel not found in transport stream"), and dispatch the corresponding DOM event. If, following this procedure, the OITF selects a tuner that was not already being used to display video inside the video/broadcast object, the OITF shall claim the selected tuner and the associated resources (e.g., decoding and rendering resources) on behalf of the video/broadcast object. If all of the following are true: • the video/broadcast object is successfully switched to the new channel, • the channel is a locally defined channel (created using the createChannelObject method), • the new channel has the same tuning parameters as a channel already in the channel list in the OITF, • the idType is a value other than ID_IPTV_URI, then the result of this operation shall be the same as calling setChannel with the channel argument being the corresponding channel object in the channel list, such that: • the values of the properties of the video/broadcast object currentChannel shall be the same as those of the channel in the channel list, • any subsequent call to nextChannel or prevChannel shall switch the tuner to the next or previous channel in the favourite list or channel list as appropriate, as described in the definitions of these methods.

	Otherwise, if any of the above conditions is not true, then: - the values of the properties of the video/broadcast object <code>currentChannel</code> shall be the same as those provided in the channel argument to this method, updated as defined in 8.4.3; - the channel is not considered to be part of the channel list. The resulting current channel after any subsequent call to <code>nextChannel()</code> or <code>prevChannel()</code> is implementation dependent, however all appropriate functions shall be called and DOM events dispatched. The OITF shall visualize the video content received over the tuner in the area of the browser allocated for the video/broadcast object. If the OITF cannot visualize the video content following a successful tuner switch (e.g. because the channel is under parental lock), the OITF shall trigger the function specified by the <code>onChannelChangeError</code> property with the appropriate <code>channel</code> and <code>errorState</code> value, and dispatch a corresponding DOM event (see below). If successful, the OITF shall trigger the function specified by the <code>onChannelChangeSucceeded</code> property with the given <code>channel</code> value, and also dispatch a corresponding DOM event.	
Arguments	channel	The channel to which a switch is requested. If the <code>channel</code> object specifies a <code>ccid</code>, the <code>ccid</code> identifies the channel to be set. If the channel does not specify a <code>ccid</code>, the <code>idType</code> determines which properties of the channel are used to define the channel to be set, for example, if the channel is of type <code>ID_IPTV_SDS</code> or <code>ID_IPTV_URI</code>, the <code>ipBroadcastID</code> identifies the channel to be set. If null, the video/broadcast object shall transition to the unrealized state and release any resources used for decoding video and/or audio. A <code>ChannelChangeSucceeded</code> event shall be generated when the operation has completed.
	trickplay	Optional flag indicating whether resources should be allocated to support trick play. This argument provides a hint to the receiver in order that it may allocate appropriate resources. Failure to allocate appropriate resources, due to a resource conflict, a lack of trickplay support, or due to the OITF ignoring this hint, shall have no effect on the success or failure of this method. If trickplay is not supported, this shall be indicated through the failure of later calls to methods invoking trickplay functionality. The <code>timeShiftMode</code> property defined in 7.13.3.3 shall provide information as to type of trickplay resources that should be allocated. If argument <code>contentAccessDescriptorURL</code> is included then the <code>trickplay</code> argument shall be included.
	contentAccessDescriptorURL	Optional argument containing a Content Access Streaming descriptor (the format of which is defined in Annex C.2) that can be included to provide additional information for dealing with IPTV broadcasts that are (partially) DRM-protected. The argument shall be <code>undefined</code> or <code>null</code> if it is not applicable.

void prevChannel()	
Description	Requests the OITF to switch the tuner that is currently in use by the video/broadcast object to the channel that precedes the current channel in the active favourite list, or, if no favourite list is currently selected, to the previous channel in the channel list. If it has reached the start of the favourite/channel list, it shall cycle to the last channel in the list. If the current channel is not part of the channel list, it is implementation-dependent whether the method call succeeds or fails and, if it succeeds, which channel is selected. In both cases, all appropriate functions shall be called and DOM events dispatched. If the previous channel is a channel that cannot be received over the tuner currently used by the video/broadcast object, the OITF shall relay the channel switch request to a local physical or logical tuner that is not in use and that can tune to the specified channel. The behaviour is defined in more detail in the description of the setChannel method. If an error occurs during switching to the previous channel, the OITF shall trigger the function specified by the onChannelChangeError property with the appropriate channel and errorState value, and dispatch the corresponding DOM event (see below). If the OITF does not maintain the channel list and favourite list by itself, the request shall fail and the OITF shall trigger the onChannelChangeError function with the channel property having the value null, and errorState=10 ("channel cannot be changed by nextChannel()/prevChannel() methods"). If successful, the OITF shall trigger the function specified by the onChannelChangeSucceeded property with the appropriate channel value, and also dispatch the corresponding DOM event. Calls to this method are valid in the Connecting, Presenting and Stopped states. They are not valid in the Unrealized state and shall fail.

void nextChannel()	
Description	Requests the OITF to switch the tuner that is currently in use by the video/broadcast object to the channel that succeeds the current channel in the active favourites list, or, if no favourite list is currently selected, to the next channel in the channel list. If it has reached the end of the favourite/channel list, it shall cycle to the first channel in the list. If the current channel is not part of the channel list, it is implementation dependent whether the method call succeeds or fails and, if it succeeds, which channel is selected. In both cases, all appropriate functions shall be called and DOM events dispatched. If the next channel is channel that cannot be received over the tuner currently used by the video/broadcast object, the OITF shall relay the channel switch request to a local physical or logical tuner that is not in use and that can tune to the specified channel. The behaviour is defined in more detail in the description of the setChannel method. If an error occurs during switching to the next channel, the OITF shall trigger the function specified by the onChannelChangeError property with the appropriate channel and errorState value, and dispatch the corresponding DOM event (see below). If the OITF does not maintain the channel list and favourite list by itself, the request shall fail and the OITF shall trigger the onChannelChangeError function with the channel property having the value null, and errorState=10 ("channel cannot be changed by nextChannel()/prevChannel() methods"). If successful, the OITF shall trigger the function specified by the onChannelChangeSucceeded property with the appropriate channel value, and also dispatch the corresponding DOM event. Calls to this method are valid in the Connecting, Presenting and Stopped states. They are not valid in the Unrealized state and shall fail.

void setFullScreen(Boolean fullscreen)		
Description	Sets the rendering of the video content to full-screen (fullscreen = true) or windowed (fullscreen = false) mode (as per [Req. 5.7.1.c] of CEA-2014-A). If this indicates a change in mode, this shall result in a change of the value of property fullScreen. Changing the mode shall not affect the z-index of the video object.	
Arguments	fullScreen	Boolean to indicate whether video content should be rendered full-screen or not.

Boolean setVolume(Integer volume)		
Description	Adjusts the volume of the currently playing media to the volume as indicated by volume. Allowed values for the volume argument are all the integer values starting with 0 up to and including 100. A value of 0 means the sound will be muted. A value of 100 means that the volume will become equal to current "master" volume of the device, whereby the "master" volume of the device is the volume currently set for the main audio output mixer of the device. All values between 0 and 100 define a linear increase of the volume as a percentage of the current master volume, whereby the OITF shall map it to the closest volume level supported by the platform. The method returns true if the volume has changed. Returns false if the volume has not changed. Applications may use the getVolume() method to retrieve the actual volume set.	
Arguments	volume	Integer value between 0 up to and including 100 to indicate volume level.

Integer getVolume()	
Description	Returns the actual volume level set; for systems that do not support individual volume control of players, this method will have no effect and will always return 100.

void release()	
Description	Releases the decoder/tuner used for displaying the video broadcast inside the video/broadcast object, stopping any form of visualization of the video inside the video/broadcast object and releasing any other associated resources. If the object was created with an allocationMethod of STATIC_ALLOCATION, the releasing of resources shall change this to DYNAMIC_ALLOCATION.

void stop()	
Description	Stop presenting broadcast video. If the video/broadcast object is in any state other than the unrealized state, it shall transition to the stopped state and stop video and audio presentation. This shall have no effect on access to non-media broadcast resources such as EIT information. Calling this method from the unrealized state shall have no effect. See 7.13.2.2 for more information of its usage.

IECNORM.COM : Click to view the full PDF of IEC 62766-5-1:2017

7.13.2.5 Events

For the intrinsic events listed in the table below, a corresponding DOM event shall be generated in the following manner:

Intrinsic event	Corresponding DOM event	DOM Event properties
onfocus	f ocus (as defined in 5.2.1.2 of the DOM Level 3 Events specification as referenced in IEC 62766-5-2)	Bubbles: No Cancellable: No Context Info: None
onblur	b lur (as defined in 5.2.1.2 of the DOM Level 3 Events specification as referenced in IEC 62766-5-2)	Bubbles: No Cancellable: No Context Info: None
onFullScreenChange	FullScreenChange	Bubbles: No Cancellable: No Context Info: None
onChannelChangeError	ChannelChangeError	Bubbles: No Cancellable: No Context Info: channel , errorState
onChannelChangeSucceeded	ChannelChangeSucceeded	Bubbles: No Cancellable: No Context Info: channel
onPlayStateChange	PlayStateChange	Bubbles: No Cancellable: No Context Info: state , error

These DOM events are directly dispatched to the event target, and will not bubble nor capture. Applications should not rely on receiving these events during the bubbling or the capturing phase. Applications that use DOM event handlers shall call the `addEventListener()` method on the video/broadcast object itself. The third parameter of `addEventListener`, i.e. "useCapture", will be ignored.

7.13.2.6 Styling

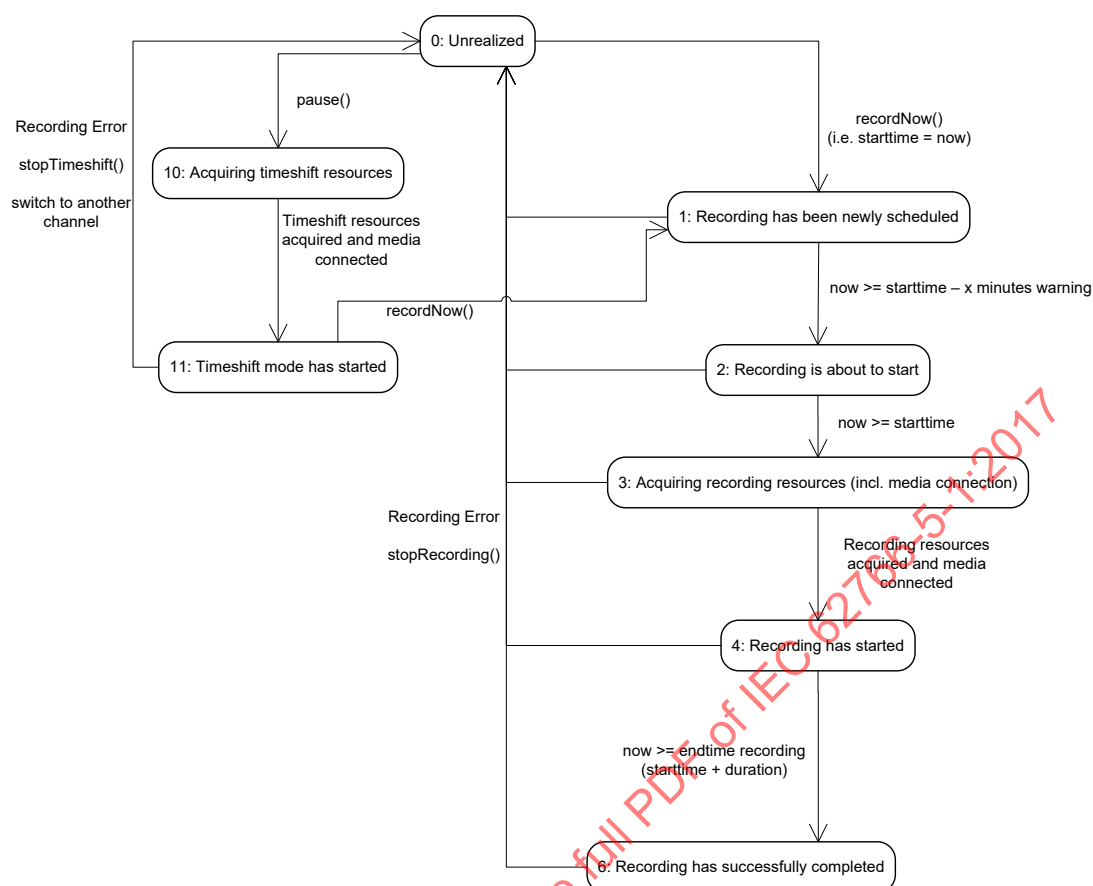
video/broadcast objects shall support CSS-property z-index, in both full-screen and windowed mode.

7.13.3 Extensions to video/broadcast for recording and time-shift

7.13.3.1 General

If an OITF has indicated support for recording functionality (i.e. by giving value `true` to element `<recording>` as specified in 9.3.4 in its capability description), the OITF shall support the following additional constants, properties and methods on the video/broadcast object, in order to start a recording and/or time-shift of a current broadcast.

This functionality is subject to the security model as specified in 10.1 and to the state transitions represented in the state diagram in Figure 15.



IEC

Figure 15 – PVR States for recordNow and timeshifting using video/broadcast

When the user switches to another channel whilst the current channel is being recorded using recordNow or the video/broadcast object gets destroyed, the conflict resolution and the release of resources is implementation dependent. The OITF may report a recording error using a RecordingEvent with value 0 ("Unrealized") for argument state and with value 2 ("Tuner conflict") for argument error in that case.

7.13.3.2 Constants

Name	Value	Use
POSITION_START	0	Indicates a playback position relative to the start of the buffered content.
POSITION_CURRENT	1	Indicates a playback position relative to the current playback position.
POSITION_END	2	Indicates a playback position relative to the end of the buffered content.

7.13.3.3 Properties

function onPlaySpeedChanged(Number speed) The function that is called when the playback speed of a channel changes. The specified function is called with one argument, speed , which is defined as follows: Number speed – the playback speed of the media at the time the event was dispatched. If the playback reaches the beginning of the time-shift buffer at rewind playback speed, then the play state is changed to 2 ('paused') and a PlaySpeedChanged event with a speed of 0 is generated. If the playback reaches the end of the time-shift buffer at fast-forward playback speed, then the play speed is set to 1,0 and a PlaySpeedChanged event is generated.

function **onPlayPositionChanged**(Integer position)

The function that is called when change occurs in the play position of a channel due to the use of trick play functions.

The specified function is called with one argument, **position**, which is defined as follows:

- **Integer position** – the playback position of the media at the time the event was dispatched, measured from the start of the timeshift buffer. If the value of the **currentTimeshiftMode** property is 1, this is measured in milliseconds from the start of the timeshift buffer. If the value of the **currentTimeshiftMode** property is 2, this is measured in milliseconds from the start of the media item. If the play position cannot be determined, this argument takes the value **undefined**.

readonly Integer **playbackOffset**

Returns the playback position, specified as the positive offset of the live broadcast in seconds, in the currently rendered (timeshifted) broadcast.

When the **currentTimeshiftMode** property has the value 1, the value of this property is **undefined**.

readonly Integer **maxOffset**

Returns the maximum playback offset, in seconds of the live broadcast, which is supported for the currently rendered (timeshifted) broadcast. If the maximum offset is unknown, the value of this property shall be **undefined**.

When the **currentTimeshiftMode** property has the value 1, the value of this property is **undefined**.

readonly Integer **recordingState**

Returns the state of the OITF's timeshift and **recordNow** functionality for the channel shown in the **video/broadcast** object. One of:

Value	Description
0	Unrealized: user/application has not requested timeshift or recordNow functionality for the channel shown. No timeshift or recording resources are claimed in this state.
1	Recording has been newly scheduled.
2	Recording is about to start. The receiver may be monitoring EPG data in order to ensure that the programme scheduled to be recorded has not been moved, or to support "accurate recording" functionality as defined in Clause 11 of ETSI TS 102 323, where slight changes in the start time of the recording do not result in the start of the recording being missed. No recording resources have yet been acquired, although the OITF may have tuned to the channel which is to be recorded.
3	Acquiring recording resources (incl. media connection).
4	Recording has started.
5	Recording has been updated.
6	Recording has successfully completed.
10	Acquiring timeshift resources (incl. media connection).
11	Timeshift mode has started.

function onRecordingEvent(Integer state, Integer error, String recordingId)

This function is the DOM 0 event handler for notification of state changes of the recording functionality. The specified function is called with the following arguments:

- **Integer state** – The current state of the recording. One of:

Value	Description
0	Unrealized: user/application has not requested timeshift or recordNow functionality for the channel shown. No timeshift or recording resources are claimed in this state.
1	Recording has been newly scheduled.
2	Recording is about to start . The receiver may be monitoring EPG data in order to ensure that the programme scheduled to be recorded has not been moved, or to support "accurate recording" functionality as defined in Clause 11 of ETSI TS 102 323, where slight changes in the start time of the recording do not result in the start of the recording being missed. No recording resources have yet been acquired, although the OITF may have tuned to the channel which is to be recorded.
3	Acquiring recording resources (incl. media connection).
4	Recording has started.
5	Recording has been updated.
6	Recording has successfully completed.
10	Acquiring timeshift resources (incl. media connection).
11	Timeshift mode has started.

- **Integer error** – If the state of the recording has changed due to an error, this field contains an error code detailing the type of error. One of:

Value	Description
0	The recording sub-system is unable to record due to resource limitations.
1	There is insufficient storage space available (some of the recording may be available).
2	Tuner conflict (e.g. due to conflicting scheduled recording).
3	Recording not allowed due to DRM restrictions.
4	Recording has stopped before completion due to unknown (probably hardware) failure.
10	Timeshift not possible due to resource limitations.
11	Timeshift not allowed due to DRM restrictions.
12	Timeshift ended due to unknown failure.

If no error has occurred, this argument shall take the value **undefined**.

- **String recordingId** – The identifier of the recording to which this event refers. This shall be equal to the value of the id property for the affected recording, if the event is associated with a specific recording.

readonly Integer playPosition

If the value of the **currentTimeshiftMode** property is 1, the current playback position of the media, measured in milliseconds from the start of the timeshift buffer.

If the value of the **currentTimeshiftMode** property is 2, the current playback position of the media, measured in milliseconds from the start of the media item.

readonly Number playSpeed

The current play speed of the media.

readonly Number **playSpeeds**[]

Returns the ordered list of playback speeds, expressed as values relative to the normal playback speed (1,0), at which the currently specified A/V content can be played (as a time-shifted broadcast in the video/broadcast object), or **undefined** if the supported playback speeds are not known or the video/broadcast object is not in timeshift mode.

If the video/broadcast object is in timeshift mode, the playSpeeds array shall always include at least values 1,0 and 0,0.

function **onplaySpeedsArrayChanged()**

The function that is called when the playSpeeds array values have changed. An application that makes use of the playSpeeds array needs to read the values of the playSpeeds property again.

Integer **timeShiftMode**

The time shift mode indicates the mode of operation for support of timeshift playback in the video/broadcast object. Valid values are:

Value	Description
0	Timeshift is turned off.
1	Timeshift shall use "local resource".
2	Timeshift shall use "network resources".
3	Timeshift shall first use "local resource" when available and fallback to "network resources".

If property is not set the default value of the property is according to preferredTimeShiftMode in 7.3.3.2.

readonly Integer **currentTimeShiftMode**

When timeshift is in operation, the property indicates which resources are currently being used. Valid values are:

Value	Description
0	No timeshift.
1	Timeshift using "local resource".
2	Timeshift using "network resources".

7.13.3.4 Methods

String recordNow (Integer duration)		
Description	Starts recording the broadcast currently rendered in the video/broadcast object. If the OITF has buffered the broadcasted content, the recording starts from the current playback position in the buffer, otherwise starts recording the broadcast stream as soon as possible after the recording resources have been acquired. The specified duration is used by the OITF to determine the minimum duration of the recording in seconds from the current starting point. Calling recordNow() while the broadcast that is currently rendered in the video/broadcast object is already being recorded, shall have no effect on the recording and shall return the value null. In other cases, this method returns a String value representing a unique identifier to identify the recording. If the OITF provides recording management functionality through the APIs defined in 7.10.5, this shall be the value of the id property of the associated Recording object defined in 7.10.6. The OITF shall guarantee that recording identifiers are unique in relation to download identifiers and CODASSET identifiers. The method returns undefined if the given argument is not accepted to trigger a recording. If the OITF supports metadata processing in the terminal, the fields of the resulting Recording object may be populated using metadata retrieved by the terminal. Otherwise, the values of these fields shall be implementation-dependent.	
Arguments	duration	The minimum duration of the recording in seconds. A value of -1 indicates that the recording should continue until stopRecording() is called, storage space is exhausted, or an error occurs. In this case it is essential that stopRecording() is called later.

void stopRecording ()		
Description	Stops the current recording started by recordNow() .	

Boolean pause ()		
Description	Pause playback of the broadcast. The action taken depends on the value of the timeShiftMode property. If the value of the timeShiftMode property is 0, if trick play is not supported for the channel currently being rendered, or if the current time shift mode is not supported for the type of channel being presented (e.g. attempting to use network resource to time shift a DVB or analogue channel) this method shall return false. If the timeshift mode is set to 1 or 3 (local resources) and if recording has not yet been started, this method will start recording the broadcast that is currently being rendered live (i.e., not time-shifted) in the video/broadcast object. If the OITF has buffered the 'live' broadcast content, the recording starts with the content that is currently being rendering in the video/broadcast object. If the recording started successfully, the rendering of the broadcast content is paused, i.e. a still-image video frame is shown. If the timeshift mode is set to 2 (network resources), then the OITF shall follow the procedures defined in 8.2.7.5 and returns true. Since this operation is asynchronous when the procedures are executed successfully, the rendering of the broadcasted content is paused, i.e. a still-image video frame is shown, and PlaySpeedChanged event is generated. If the specified timeshift mode is not supported, this method shall return false. Otherwise, this method shall return true. Acquiring the necessary resources to enable the specified timeshift mode may be an asynchronous operation; applications may receive updates of this process by registering a listener for RecordingEvents as defined in 7.13.2.5. If trick play is not supported for the channel currently being rendered, this method shall return false, otherwise true is returned. This operation may be asynchronous, and presentation of the video may not pause until after this method returns. For this reason, a PlaySpeedChanged event will be generated when the operation has completed, regardless of the success of the operation. If the operation fails, the argument of the event shall be set to the previous play speed.	

	Boolean resume()
Description	Resumes playback of the time-shifted broadcast channel that is currently being rendered in the video/broadcast object at the speed specified by setSpeed(). If the desired speed was not set via setSpeed(), playback is resumed at normal speed (i.e. speed 1,0). If the video/broadcast object is currently not rendering a time-shifted channel, the OITF shall ignore the request to start playback and shall return false. If playback cannot be resumed, the OITF shall also return false, otherwise true is returned. This operation may be asynchronous, and presentation of the video may not resume until after this method returns. For this reason, a PlaySpeedChanged event will be generated when the operation has completed, regardless of the success of the operation. If the operation fails, the argument of the event shall be set to the previous play speed. The action taken depends on the value of the timeShiftMode property. If the value of the timeShiftMode property is 1 or 3 (local resources), then the OITF shall resume playback of the broadcast channel as specified above and return true. If the value of the timeShiftMode property is 2 (network resources), then the OITF shall follow the procedures defined in 8.2.7.5 and return true. Since this operation is asynchronous when the procedure is successfully executed a PlaySpeedChanged event is generated with current speed. After initial operation of resume(), several events may affect the operation. If during fast forward the end of the stream is reached, the playback shall resume at normal speed and a PlaySpeedChanged event is generated. If the end of stream is reached due to end of content, the playback will automatically be paused and a PlaySpeedChanged event is generated. Any resources used for time-shifting shall not be discarded. If during rewinding, the playback reaches the point that it cannot be rewound further, playback will automatically be paused (i.e. the play speed will be changed to 0) and a PlaySpeedChanged event is generated. If for any of these events timeShiftMode is set to 3 and local resources are not available anymore, then network sources shall be used according to the procedures defined in 8.2.7.5. The OITF shall perform a smooth transition of the stream between local and network resources.

Boolean setSpeed (Number speed)		
Description		Sets the playback speed of the time-shifted broadcast to the value speed, without changing the paused/resumed state of the time-shifted broadcast. When playback is paused (i.e. by setting the play speed to 0), the last decoded frame of video is displayed. If the time-shifted broadcast cannot be played at the desired speed, specified as a value relative to the normal playback speed, the playback speed will be set to the best approximation of speed. Applications are not required to pause playback of the broadcast or take any other action before calling setSpeed(). If the video/broadcast object is currently not rendering a time-shifted channel, the OITF shall ignore the request to change the playback speed and shall return false, otherwise true is returned. This operation may be asynchronous, and presentation of the video may not be affected until after this method returns. For this reason, a PlaySpeedChanged event will be generated when the operation has completed, regardless of the success of the operation. If the operation fails, the argument of the event shall be set to the previous play speed. The action taken depends on the value of the timeShiftMode property. If the value of the timeShiftMode property is 1 or 3 (local resources), then the setSpeed() method sets the playback speed of the time-shifted broadcast to the value speed. If the timeShiftMode is set to 2 (network resources), the OITF shall follow the procedures defined in 8.2.7.5 and return true. Since this operation is asynchronous when the procedure is successfully executed PlaySpeedChanged event is generated with the new speed. After initial operation of setSpeed() several events may affect the operation. If during fast forward the end of stream is reached the playback shall resume at normal speed and a PlaySpeedChanged event is generated. If the end of stream is reached due to end of content the playback will automatically be paused and a PlaySpeedChanged event is generated. Any resources used for time-shifting shall not be discarded. If during rewinding the playback has reaches the point that it cannot be rewound further, playback shall resume at normal speed and a PlaySpeedChanged event is generated. If for any of these events if timeShiftMode is set to 3 and local resources are not available anymore, then network sources shall be used according to the procedures defined in 8.2.7.5. The OITF shall perform a smooth transition of the stream between local and network resources.
Arguments	speed	The desired relative playback speed, specified as a float value relative to the normal playback speed of 1.0. A negative value indicates reverse playback. If the time-shifted broadcast cannot be played at the desired speed, the playback speed will be set to the best approximation.

Boolean seek(Integer offset, Integer reference)		
Description	Sets the playback position of the time-shifted broadcast that is being rendered in the video/broadcast object to the position specified by the offset and the reference point as specified by one of the constants defined in 7.13.3.2. Returns true if the playback position is a valid position to seek to, false otherwise. If the video/broadcast object is currently not rendering a time-shifted channel or if the position falls outside the time-shift buffer, the OITF shall ignore the request to seek and shall return the value false. Applications are not required to pause playback of the broadcast or take any other action before calling seek(). This operation may be asynchronous, and presentation of the video may not be affected until after this method returns. For this reason, a PlayPositionChanged event will be generated when the operation has completed, regardless of the success of the operation. If the operation fails, the argument of the event shall be set to the previous play position. The action taken depends on the value of the timeShiftMode property. If the timeShiftMode is set to 1 (local resources), the seek() method sets the playback position of the time-shifted broadcast that is being rendered in the video/broadcast object as defined above. Playback of live content is resumed if the new position equals the end of the time-shift buffer. If the timeShiftMode is set to 2 (network resources) the OITF shall follow the procedures defined in 8.2.7.5 and return true. Since this operation is asynchronous when the procedure is successfully executed, PlayPositionChanged event is generated with the new position. Note that if timeShiftMode is set to 3, then local resources are used over network resources. After initial operation of seek() several events may affect the operation. If during fastforward the end of stream is reached the playback shall resume at normal speed and a PlaySpeedChanged event is generated. If the end of stream is reached due to end of content the playback will automatically be paused and a PlaySpeedChanged event is generated. Any resources used for time-shifting shall not be discarded. If for any of these events if timeShiftMode is set to 3 and local resources are not available anymore, then network sources shall be used according to the procedures defined in 8.2.7.5. The OITF shall perform a smooth transition of the stream between local and network resources.	
Arguments	offset	The offset from the reference position, in seconds. This can be either a positive value to indicate a time later than the reference position or a negative value to indicate a time earlier than the reference position.
	reference	The reference point from which the offset shall be measured. The reference point can be either POSITION_CURRENT , POSITION_START , or POSITION_END .

Boolean stopTimeshift()	
Description	Stops rendering in time-shifted mode of the broadcast channel in the video/broadcast object and, if applicable, plays the current broadcast from the live point and stops time-shifting the broadcast. The OITF shall release all resources that were used to support time-shifted rendering of the broadcast. Returns true if the time-shifted broadcast was successfully stopped and resources were released and false otherwise. If the video/broadcast object is currently not rendering a time-shifted channel, the OITF shall ignore the request to stop the time-shift and shall return the value false.

In addition to these methods, the OITF shall support an additional optional attribute “offset” on the **setChannel(Channel channel, Boolean trickplay, String contentAccessDescriptorURL)** method of the **video/broadcast** object as defined in 7.13.2.4, if the OITF has indicated support for scheduled content over IP by defining one or more **ID_IPTV_*** values as part of the transport attribute of the **<video_broadcast>** element in the capability description.

void setChannel(Channel channel, Boolean trickplay, String contentAccessDescriptorURL, Integer offset)		
Description	Requests the OITF to switch a (logical or physical) tuner to the specified channel and render the received broadcast content in the area of the browser allocated for the video/broadcast object, as specified by the setChannel(Channel channel, Boolean trickPlay, String contentAccessDescriptorURL) method in 7.13.2.4. The additional offset attribute optionally specifies the desired offset with respect to the live broadcast in number of seconds from which the OITF should start playback immediately after the channel switch (whereby offset is given as a positive value for seeking to a time in the past). If an OITF cannot start playback from the desired position, as indicated by the specified offset (e.g. because the OITF did not, or could not, record the specified channel prior to the call to setChannel), if the specified offset is '0', or if the offset is not specified, the OITF shall start playback from the live position after the specified channel switch.	
Arguments	channel	As defined for method setChannel() in 7.13.2.4.
	trickplay	Optional flag as defined for method setChannel() in 7.13.2.4.
	contentAccessDescriptorURL	Optional attribute as defined for method setChannel() in 7.13.2.4.
	offset	The optional offset attribute may be used to specify the desired offset with respect to the live broadcast in number of seconds from which the OITF should start playback immediately after the channel switch (whereby offset is given as a negative value for seeking to a time in the past).

7.13.3.5 Events

For the intrinsic events listed in the table below, a corresponding DOM event shall be generated, in the following manner:

Intrinsic event	Corresponding DOM event	DOM Event properties
onRecordingEvent	RecordingEvent	Bubbles: No Cancellable: No Context Info: state, error, recordingId
onPlaySpeedChanged	PlaySpeedChanged	Bubbles: No Cancellable: No Context Info: speed
onPlayPositionChanged	PlayPositionChanged	Bubbles: No Cancellable: No Context Info: position
onPlaySpeedsArrayChanged	PlaySpeedsArrayChanged	Bubbles: No Cancellable: No Context Info: None

The DOM events are directly dispatched to the event target, and will not bubble nor capture. Applications should not rely on receiving these events during the bubbling or the capturing phase. Applications that use DOM event handlers shall call the `addEventListener()` method on the video/broadcast object itself. The third parameter of `addEventListener`, i.e. "useCapture", will be ignored.

7.13.4 Extensions to video/broadcast for access to EIT p/f

7.13.4.1 General

The following properties and events shall be added to the video/broadcast embedded object, if the OITF has indicated support for accessing DVB-SI EIT p/f information, by giving the

value "true" to element <clientMetadata> and the value "eit-pf" or "dvd-si" to the type attribute of that element as defined in 9.3.8 in their capability profile.

Access to these properties shall adhere to the security model in Clause 10. The associated permission name is "permission_metadata".

readonly ProgrammeCollection programmes
The collection of programmes available on the currently tuned channel. This list is a ProgrammeCollection as defined in 7.16.3 and is ordered by start time, so index 0 will always refer to the present programme (if this information is available). If the type attribute of the <clientMetadata> element in the OITF's capability description has the value "eit-pf", this list shall at least provide Programme objects as defined in 7.16.2 for the present and the directly following programme on the currently tuned channel, if that information is available. In other words, the DAE application should not expect programmes.length to be larger than 2. If the video/broadcast object is not currently tuned to a channel, or if the present/following information has not yet been retrieved (e.g. the object has just tuned to a new channel and present/following information has not yet been broadcast), or if present/following information is not available for the current channel, the length of this collection shall be 0. If the type attribute of the <clientMetadata> element in the OITF's capability description has a value other than "eit-pf", an OITF may populate this field from other metadata sources described in IEC 62766-3. The programmes.length property shall indicate the number of items that are currently known and up to date (i.e. whereby the "startTime + duration" is not smaller than the current time). This may be 0 if no programme information is currently known for the currently tuned channel. In order to prevent misuse of this information, access to this property shall adhere to the security model in Clause 10. The associated permission name is "permission_metadata".

function onProgrammesChanged()
The function that is called when the programmes property has been updated with new programme information, e.g. when the current broadcast programme is finished and a new one has started. The specified function is called with no arguments.

7.13.4.2 Events

For the intrinsic events listed in the table below, a corresponding DOM event shall be generated in the following manner:

Intrinsic event	Corresponding DOM event	DOM Event properties
onProgrammesChanged	ProgrammesChanged	Bubbles: No Cancellable: No Context Info: None

This DOM event is directly dispatched to the event target, and will not bubble nor capture. Applications should not rely on receiving these events during the bubbling or the capturing phase. Applications that use DOM event handlers shall call the `addEventListener()` method on the **video/broadcast** object itself. The third parameter of `addEventListener`, i.e. "useCapture", will be ignored.

7.13.5 Extensions to video/broadcast for playback of selected components

To support the selection of specific A/V components for playback (e.g. a specific subtitle language, audio language, or camera angle), the classes defined in 7.16.5.3 to 7.16.5.6 shall be supported and the constants, properties and methods defined in 7.16.5.2 shall be supported on the **video/broadcast** object.

7.13.6 Extensions to video/broadcast for parental ratings errors

7.13.6.1 General

For parental rating related errors or changes during playback of A/V content through the video/broadcast object, an OITF shall support the following intrinsic event properties and corresponding DOM events for the video/broadcast object:

function	onParentalRatingChange (String contentID, ParentalRatingCollection ratings, String DRMSystemID, Boolean blocked)
The function that is called whenever the parental rating of the content being played inside the embedded object changes. These events may occur at the start of a new content item, or during playback of a content item (e.g. during playback of linear TV content). The specified function is called with four arguments contentID, rating, DRMSystemID and blocked which are defined as follows: • String contentID – the content ID to which the parental rating change applies. If the event is generated by the DRM system, it shall be the unique identifier for that content in the context of the DRM system (i.e. in the case of Marlin BB it is the Marlin contentID, in the case of CSPG-CI+ the value of this field is null). Otherwise it may be null or undefined. • ParentalRatingCollection ratings – the parental ratings of the currently playing content. The ParentalRatingCollection object is defined in 7.9. • String DRMSystemID – the DRM System ID of the DRM system that generated the event as defined by element DRMSystemID in 4.3.3 of IEC 62766-3:2016. The value shall be null if the parental control is not enforced by a particular DRM system. • Boolean blocked – flag indicating whether consumption of the content is blocked by the parental control system as a result of the new parental rating value.	

function	onParentalRatingError (String contentID, ParentalRatingCollection ratings, String DRMSystemID)
The function that is called when a parental rating error occurs during playback of A/V content inside the embedded object, and is triggered whenever one or more parental ratings are discovered and none of them are valid. A valid parental rating is defined as one which uses a parental rating scheme that is supported by the OITF and which has a parental rating value that is supported by the OITF. The specified function is called with three arguments contentID, rating, and DRMSystemID which are defined as follows: • String contentID – the content ID to which the parental rating error applies. If the event is generated by the DRM system, it shall be the unique identifier for that content in the context of the DRM system (i.e. in the case of Marlin BB it is the Marlin contentID, in the case of CSPG-CI+ the value of this field is null). Otherwise it may be null or undefined. • ParentalRatingCollection ratings – the parental ratings of the currently playing content. The ParentalRatingCollection object is defined in 7.9. • String DRMSystemID – optional argument that specifies the DRM System ID of the DRM system that generated the event as defined by element DRMSystemID in 4.4.3 of IEC 62766-3:2016. The value shall be null if the parental control is not enforced by a particular DRM system.	

7.13.6.2 Events

For the intrinsic events listed in the table below, a corresponding DOM event shall be generated, in the following manner:

Intrinsic event	Corresponding DOM event	DOM Event properties
onParentalRatingChange	ParentalRatingChange	Bubbles: No Cancellable: No Context Info: contentID, ratings, DRMSystemID, blocked
onParentalRatingError	ParentalRatingError	Bubbles: No Cancellable: No Context Info: contentID, ratings, DRMSystemID

The above DOM events are directly dispatched to the event target, and will not bubble nor capture. Applications should not rely on receiving a ParentalRatingError event during the bubbling or the capturing phase. The Applications that use DOM event handlers shall call the addEventListener() method on the video/broadcast object itself. The third parameter of addEventListener, i.e. "useCapture", will be ignored.

7.13.7 Extensions to video/broadcast for DRM rights errors

This subclause shall apply to OITF and/or server devices which have indicated support for DRM protection by providing one or more <drm> elements as specified in 9.3.11.

For notifying JavaScript about DRM licensing errors during playback of DRM protected A/V content through the "video/broadcast" object, an OITF shall support the following intrinsic event property and corresponding DOM event for the "video/broadcast" object:

<pre>function onDRMRightsError(Integer errorState, String contentID, String DRMSystemID, String rightsIssuerURL)</pre>
The function that is called: - whenever a rights error occurs for the A/V content (no licence, licence invalid), which has led to blocking consumption of the content; - whenever a rights change occurs for the A/V content (licence valid), which leads to unblocking the consumption of the content. This may occur during playback, recording or timeshifting of DRM protected AV content. The specified function is called with four arguments errorState, contentID, DRMSystemID and rightsIssuerURL which are defined as follows: - Integer errorState – error code detailing the type of error: 0: no licence, consumption of the content is blocked. 1: invalid licence, consumption of the content is blocked. 2: valid licence, consumption of the content is unblocked. - String contentID – the unique identifier of the protected content in the scope of the DRM system that raises the error (i.e. in the case of Marlin BB it is the Marlin contentID, in the case of CSPG-CI+ the value of this field is null). - String DRMSystemID – DRMSystemID as defined by element DRMSystemID in 4.4.3 of IEC 62766-3:2016. For example, for Marlin, the DRMSystemID value is "urn:dvb:casystemid:19188". - String rightsIssuerURL – optional element indicating the value of the rightsIssuerURL that can be used to non-silently obtain the rights for the content item currently being played for which this DRM error is generated, in cases whereby the rightsIssuerURL is known. Cases whereby the rightsIssuerURL is known include cases whereby the rightsIssuerURL has been extracted from the MPEG2_TS of the protected content, retrieved from the SD&S discovery record or from the associated BCG metadata. The corresponding rightsIssuerURL fields are defined in 4.2.2.4 of IEC 62766-7:2017 and in 4.4.3 of IEC 62766-3:2016 respectively. If different URLs are retrieved from the stream and the metadata, then the conflict resolution is implementation-dependent.

For the intrinsic events listed in the table below, a corresponding DOM event shall be generated, in the following manner:

Intrinsic event	Corresponding DOM event	DOM Event properties
onDRMRightsError	DRMRightsError	Bubbles: No Cancellable: No Context Info: errorState, contentID, DRMSystemID, rightsIssuerURL

The above DOM event is directly dispatched to the event target, and will not bubble nor capture. Applications should not rely on receiving a DRMRightsError event during the bubbling or the capturing phase. Applications that use DOM event handlers shall call the `addEventListener()` method on the video/broadcast object itself. The third parameter of `addEventListener`, i.e. "useCapture", will be ignored.

7.13.8 Extensions to video/broadcast for current channel information

7.13.8.1 General

If an OITF has indicated support for extended tuner control (i.e. by giving value true to element `<extendedAVControl>` as specified in 9.3.7 in its capability description), the OITF shall support the following additional properties and methods on the video/broadcast object.

The functionality as described in this subclause is subject to the security model of 10.1.4.9.

NOTE The property `onChannelScan` and methods `startScan` and `stopScan` have been moved to 7.13.10.

7.13.8.2 Properties

readonly Channel <code>currentChannel</code>
The channel currently being presented by this embedded object if the user has given permission to share this information, possibly through a mechanism outside the scope of this document. If no channel is being presented, or if this information is not visible to the caller, the value of this property shall be null.
The value of this property is not affected during timeshift operations and shall reflect the value prior to the start of a timeshift operation, for both local and network timeshift resources.

7.13.9 Extensions to video/broadcast for creating channel lists from SD&S fragments

Note that the method `createChannelList()` shall be moved to 7.13.10.

7.13.10 The ChannelConfig class

7.13.10.1 General

The `ChannelConfig` class provides the entry point for applications to get information about the list of channels available. It can be obtained in two ways:

- by calling the method `getChannelConfig()` of the video/broadcast embedded object as defined in 7.13.1.3;
- by calling the method `createChannelConfig()` of the object factory API as defined in 7.1.2.

The availability of the properties and methods are dependent on the capabilities description as specified in 9.3. The following table provides a list of the capabilities and the associated properties and methods. If the capability is false, the properties and methods shall not be available to the application. Properties and methods not listed in the following table shall be

available to all applications as long as the OITF has indicated support for tuner control (i.e. `<video_broadcast>true</video_broadcast>` as defined in 9.3.2) in their capability.

Capability	Properties	Methods
Element <code><extendedAVControl></code> is set to <code>"true"</code> as defined in 9.3.7.	<code>onChannelScan</code>	<code>startScan()</code> <code>stopScan()</code>
Element <code><video_broadcast type="ID_IPTV_SDS"></code> is set as defined in 9.3.7.		<code>createChannelList()</code>

The functionality as described in this subclause is subject to the security model of 10.1.4.2.2 (for obtaining a `ChannelConfig` object) and 10.1.4.9 (for properties and methods covered by the `<extendedAVControl>` capability as defined below).

7.13.10.2 Properties

<code>readonly ChannelList channelList</code>
The list of channels. If an OITF includes a platform-specific application that enables the end-user to choose a channel to be presented from a list then all the channels in the list offered to the user by that application shall be included in this <code>ChannelList</code>. The list of channels will be a subset of all those available to the OITF. The precise algorithm by which this subset is selected will be market and/or implementation dependent. For example: - if an OITF with a DVB-T tuner receives multiple versions of the same channel, one would be included in the list and the duplicates discarded; - an OITF with a DVB tuner will often filter services based on service type to discard those which are obviously inappropriate or impossible for that device to present to the end-user, e.g. firmware download services. The order of the channels in the list corresponds to the channel ordering as managed by the OITF shall return the value <code>null</code> if the channel list is not (partially) managed by the OITF (i.e., if the channel list information is managed entirely in the network). The properties of channels making up the channel list shall be set by the OITF to the appropriate values as determined by the tables in 8.4.3. The OITF shall store all these values as part of the channel list. Some values are set according to the data carried in the broadcast stream. In this case, the OITF may set these values to <code>undefined</code> until such time as the relevant data has been received by the OITF, for example after tuning to the channel. Once the data has been received, the OITF shall update the properties of the channel in the channel list according to the received data. NOTE There is no requirement for the OITF to pro-actively tune to every channel to gather such data.

<code>readonly FavouriteListCollection favouriteLists</code>
A list of favourite lists. shall return the value <code>null</code> if the favourite lists are not (partially) managed by the OITF (i.e. if the favourite lists information is managed entirely in the network).

<code>readonly FavouriteList currentFavouriteList</code>
Currently active favourite channel list object. If <code>currentFavouriteList</code> is <code>undefined</code>, no favourite filter list is currently applied. The OITF shall return the value <code>null</code> if the favourite lists are not (partially) managed by the OITF (i.e. if the favourite lists information is managed entirely in the network).

```
function onChannelScan( Integer scanEvent, Integer progress, Integer
frequency,
Integer signalStrength, Integer channelNumber, Integer
channelType,
Integer channelCount, Integer transponderCount, Channel
newChannel)
```

This function is the DOM 0 event handler for events relating to channel scanning. On IP-only receivers, setting this property shall have no effect.

The specified function is called with the following arguments:

- Integer scanEvent – The type of event. Valid values are:

Value	Description
0	A channel scan has started.
1	Indicates the current progress of the scan.
2	A new channel has been found.
3	A new transponder has been found.
4	A channel scan has completed.
5	A channel scan has been aborted.

- Integer progress – the progress of the scan. Valid values are in the range 0 – 100, or –1 if the progress is unknown.
- Integer frequency – The frequency of the transponder in kHz (for scans on RF sources only).
- Integer signalStrength – The signal strength for the current channel. Valid values are in the range 0 – 100, or –1 if the signal strength is unknown.
- Integer channelNumber – The logical channel number of the channel that has been found.
- Integer channelType – The type of channel that has been found. Valid values are the same as for Channel.channelType.
- Integer channelCount – The total number of channels found so far during the scan.
- Integer transponderCount – The total number of transponders found so far during the scan (RF sources only).
- Channel newChannel – When scanEvent equals 2, this argument provides a reference to the Channel object that represents the newly identified channel. For other scanEvent values this argument shall be NULL.

```
function onChannelListUpdate()
```

This function is the DOM 0 event handler for events relating to channel list updates. Upon receiving a ChannelListUpdate event, if an application has references to any Channel objects then it should dispose of them and rebuild its references. Where possible, Channel objects are updated rather than removed, but their order in the ChannelConfig.all collection may have changed. Any lists created with ChannelConfig.createFilteredList() should be recreated if channels have been removed.

```
readonly Channel currentChannel
```

The current channel of the OITF if the user has given permission to share this information, possibly through a mechanism outside the scope of this document. If no channel is being presented, or if this information is not visible to the caller, the value of this property shall be null.

In an OITF where exactly one video/broadcast object is in any state other than Unrealized and the channel being presented by that video/broadcast object is the only broadcast channel being presented by the OITF, then changes to the channel presented by that video/broadcast object shall result in changes to the current channel of the OITF.

In an OITF that is presenting more than one broadcast channel at the same time, the current channel of the OITF is the channel whose audio is being presented (as defined in the bindToCurrentChannel() method). If that current channel is under the control of a DAE application via a video/broadcast object, then changes to the channel presented by that video/broadcast object shall result in changes to the current channel of the OITF.

7.13.10.3 Methods

ChannelList createFilteredList(Boolean blocked, Boolean favourite, Boolean hidden, String favouriteListID)										
Description	Create a filtered list of channels. Returns a subset of ChannelConfig.channelList.									
	The blocked, favourite and hidden flags indicate whether a channel is included in the returned list. These flags correspond to the properties on Channel with the same names. Each flag may be set to one of three values:									
	<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>true</td><td>The channel is added if and only if the corresponding property has the value true.</td></tr><tr><td>false</td><td>The channel is added if and only if the corresponding property has the value false.</td></tr><tr><td>undefined</td><td>The channel is added regardless of the state of the corresponding property.</td></tr></table>	Value	Meaning	true	The channel is added if and only if the corresponding property has the value true.	false	The channel is added if and only if the corresponding property has the value false.	undefined	The channel is added regardless of the state of the corresponding property.	
	Value	Meaning								
	true	The channel is added if and only if the corresponding property has the value true.								
false	The channel is added if and only if the corresponding property has the value false.									
undefined	The channel is added regardless of the state of the corresponding property.									
A channel will only be added to the list if the values of all three flags allow it to be added.										
The favouriteListID attribute is used to select a particular favouriteList that the createFilteredList method uses as a basis of the filtering process. If favouriteListID is the empty string (i.e. ""), then the filtering is performed on all available channels as defined by ChannelConfig.channelList.										
Arguments	blocked	Flag indicating whether manually blocked channels shall be added to the list.								
	favourite	Flag indicating whether favourite channels shall be added to the list.								
	hidden	Flag indicating whether hidden channels shall be added to the list.								
	favouriteListID	If the value of the favourite flag is true, indicates which favourites list shall be filtered upon.								

void startScan(ChannelScanOptions options, ChannelScanParameters scanParameters)		
Description	Start a scan for new channels on all available sources. When each source finishes scanning, an UpdateEvent shall be raised with the type CHANNELS_INVALIDATED and any channel lists for that source shall have been updated.	
	On IP-only receivers, this method shall have no effect.	
Arguments	options	The options to the channel scan operation.
	scanParameters	The tuning parameters to be scanned. The value of this argument shall be one of the classes that implement the ChannelScanParameters interface and shall not be an instance of the ChannelScanParameters class.

void stopScan()		
Description	Stop a channel scan, if one is in progress. Any sources that have not finished scanning shall have their scans aborted and channel line-ups for shall not be changed.	
	On IP-only receivers, this method shall have no effect.	

ChannelList createChannelList(String bdr)		
Description	Creates a ChannelList object from the specified SD&S Broadcast Discovery Record. Channels in the returned channel list will not be included in the channel list that can be retrieved via calls to getConfig().	
	An XML-encoded string containing an SD&S Broadcast Discovery Record as specified in IEC 62766-3. If the string is not a valid Broadcast Discovery Record, this method shall return null.	
Arguments	bdr	

Channel createChannelObject (Integer idType, Integer onid, Integer tsid, Integer sid, Integer sourceID, String ipBroadcastID)		
Description	Creates a Channel object of the specified idType. The Channel object can subsequently be used by the setChannel method to switch a tuner to a channel that is not part of the channel list which was conveyed by the OITF to the server. The scope of the resulting Channel object is limited to the JavaScript environment (incl. video/broadcast object) to which the Channel object is returned, i.e. it does not get added to the channellist available through method getChannelConfig. If the channel of the given idType cannot be created or the given (combination of) arguments are not considered valid or complete, the method shall return null. If the channel of the given type can be created and arguments are considered valid and complete, the method shall return a Channel object whereby at a minimum the properties with the same names are given the same value as the given arguments of the createChannelObject method. The values specified for the remaining properties of the Channel object are set to undefined.	
Arguments	idType	The type of channel, as indicated by one of the ID_* constants defined in 7.13.12.2.
	onid	The original network ID. Optional argument that shall be specified when the idType specifies a channel of type ID_DVB_* or ID_ISDB_*.
	tsid	The transport stream ID. Optional argument that may be specified when the idType specifies a channel of type ID_DVB_* or ID_ISDB_*.
	sid	The service ID. Optional argument that shall be specified when the idType specifies a channel of type ID_DVB_* or ID_ISDB_*.
	sourceID	The source ID. Optional argument that shall be specified when the idType specifies a channel of type ID_ATSC_T.
	ipBroadcastID	The DVB textual service identifier of the IP broadcast service, specified in the format "ServiceName.DomainName", or the URI of the IP broadcast service. Optional argument that shall be specified when the idType specifies a channel of type ID_IPTV_SDS or ID_IPTV_URI.

ChannelScanParameters createChannelScanParametersObject (Integer idType)		
Description	Create an instance of one of the subclasses of the ChannelScanParameters class (or one of its subclasses). The exact subclass that will be returned shall be determined by the value of the idType parameter. Initial values of all properties on the returned object shall be undefined.	
Arguments	idType	The type of object to be created. Valid values are given by the following constants on the Channel class (see 7.13.12.2): • ID_DVB_T or ID_DVB_T2 – returns an instance of the DVB_T_ChannelScanParameters class. • ID_DVB_C or ID_DVB_C2 – returns an instance of the DVB_C_ChannelScanParameters class. • ID_DVB_S or ID_DVB_S2 – returns an instance of the DVB_S_ChannelScanParameters class. • ID_ATSC_T – returns an instance of the ATSC_T_ChannelScanParameters class. All other values, or channel types which are not supported by the OITF, shall cause this method to return null.

ChannelScanOptions createChannelScanOptionsObject ()	
Description	Create an instance of the ChannelScanOptions class. Initial values of all properties on the returned object shall be undefined .

7.13.10.4 Events

For the intrinsic events listed in the table below, a corresponding DOM event shall be generated, in the following manner:

Intrinsic event	Corresponding DOM event	DOM Event properties
onChannelScan	ChannelScan	Bubbles: No Cancellable: No Context Info: scanEvent, progress, frequency, signalStrength, channelNumber, channelType, channelCount, transponderCount, newChannel
onChannelListUpdate	ChannelListUpdate	Bubbles: No Cancellable: No Context Info: None

The above DOM event is directly dispatched to the event target, and will not bubble nor capture. Applications should not rely on receiving these events during the bubbling or the capturing phase. Applications that use DOM event handlers shall call the `addEventListener()` method on the `ChannelConfig` object itself. The third parameter of `addEventListener`, i.e. "useCapture", will be ignored.

7.13.11 The ChannelList class

7.13.11.1 General

```
typedef Collection<Channel> ChannelList
```

A `ChannelList` represents a collection of `Channel` objects. See Annex I for the definition of the collection template.

In addition to the methods and properties defined for generic collections, the `ChannelList` class supports the additional properties and methods defined below.

7.13.11.2 Methods

Channel getChannel (String channelId)		
Description	Return the first channel in the list with the specified channel identifier. Returns <code>null</code> if no corresponding channel can be found.	
Arguments	channelID	The channel identifier of the channel to be retrieved, which is a value as defined for the <code>ccid</code> and <code>ipBroadcastID</code> properties of the <code>Channel</code> object as defined in 7.13.12.

Channel getChannelByTriplet (Integer onid, Integer tsid, Integer sid, Integer nid)		
Description	Return the first (IPTV or non-IPTV) channel in the list that matches the specified DVB or ISDB triplet (original network ID, transport stream ID, service ID). Where no channels of type ID_ISDB_* or ID_DVB_* are available, or no channel identified by this triplet are found, this method shall return null .	
Arguments	onid	The original network ID of the channel to be retrieved.
	tsid	The transport stream ID of the channel to be retrieved. If set to null the client shall retrieve the channel defined by the combination of onid and sid. This makes it possible to retrieve the correct channel also in case a remultiplexing took place which led to a changed tsid.
	sid	The service ID of the channel to be retrieved.
	nid	An optional argument, indicating the network id to be used select the channel when the channel list contains more than one entry with the same onid, tsid and sid.

Channel getChannelBySourceID (Integer sourceID)		
Description	Return the first (IPTV or non-IPTV) channel in the list with the specified ATSC source ID. Where no channels of type ID_ATSC_* are available, or no channel with the specified source ID is found in the channel list, this method shall return null .	
Arguments	sourceID	The ATSC source_ID of the channel to be returned.

7.13.12 The Channel class

7.13.12.1 General

The Channel object represents a broadcast stream or service.

Channel objects typically represent channels stored in the channel list (see 7.13.11). Channel objects may also represent locally defined channels created by an application using the createChannelObject() methods on the video/broadcast embedded object or the ChannelConfig class or the createChannelList() method on the ChannelConfig class. Accessing the channel property of a ScheduledRecording object or Recording object which is scheduled on a locally defined channel shall return a Channel object representing that locally defined channel.

Except for the hidden property, writing to the writable properties on a Channel object shall have no effect for Channel objects representing channels stored in the channel list. Applications should only change these writable properties of a locally defined channel before the Channel object is referenced by another object or passed to an API call as an input parameter. The effects of writing to these properties at any other time is implementation dependent.

The Channel class is defined in 7.13.12.2, 7.13.12.3, and 7.13.12.4.

7.13.12.2 Constants

Name	Value	Use
TYPE_TV	0	Used in the <code>channelType</code> property to indicate a TV channel.
TYPE_RADIO	1	Used in the <code>channelType</code> property to indicate a radio channel.
TYPE_OTHER	2	Used in the <code>channelType</code> property to indicate that the type of the channel is unknown, or known but not of type TV or radio.
TYPE_ALL	128	Used during channel scanning to indicate that all TV, radio and other channel types should be scanned.
TYPE_HBBTV_DATA	256	Reserved for data services defined by ETSI TS 102 796.
ID_ANALOG	0	Used in the <code>idType</code> property to indicate an analogue channel identified by the property 'freq' and optionally 'cni' or 'name'.
ID_DVB_C	10	Used in the <code>idType</code> property to indicate a DVB-C channel identified by the three properties: 'onid', 'tsid', 'sid'.
ID_DVB_S	11	Used in the <code>idType</code> property to indicate a DVB-S channel identified by the three properties: 'onid', 'tsid', 'sid'.
ID_DVB_T	12	Used in the <code>idType</code> property to indicate a DVB-T channel identified by the three properties: 'onid', 'tsid', 'sid'.
ID_DVB_SI_DIRECT	13	Used in the <code>idType</code> property to indicate a channel that is identified through its delivery system descriptor as defined by DVB-SI ETSI EN 300 468, 6.2.13.
ID_DVB_C2	14	Used in the <code>idType</code> property to indicate a DVB-C or DVB-C2 channel identified by the three properties: 'onid', 'tsid', 'sid'.
ID_DVB_S2	15	Used in the <code>idType</code> property to indicate a DVB-S or DVB-S2 channel identified by the three properties: 'onid', 'tsid', 'sid'.
ID_DVB_T2	16	Used in the <code>idType</code> property to indicate a DVB-T or DVB-T2 channel identified by the three properties: 'onid', 'tsid', 'sid'.
ID_ISDB_C	20	Used in the <code>idType</code> property to indicate an ISDB-C channel identified by the three properties: 'onid', 'tsid', 'sid'.
ID_ISDB_S	21	Used in the <code>idType</code> property to indicate an ISDB-S channel identified by the three properties: 'onid', 'tsid', 'sid'.
ID_ISDB_T	22	Used in the <code>idType</code> property to indicate an ISDB-T channel identified by the three properties: 'onid', 'tsid', 'sid'.
ID_ATSC_T	30	Used in the <code>idType</code> property to indicate a terrestrial ATSC channel identified by the property 'sourceID'.
ID_IPTV_SDS	40	Used in the <code>idType</code> property to indicate an IP broadcast channel identified through SD&S by a DVB textual service identifier specified in the format "ServiceName.DomainName" as value for property 'ipBroadcastID', with ServiceName and DomainName as defined in ETSI TS 102 034. This <code>idType</code> shall be used to indicate Scheduled content service defined by IEC 62766-4-1.
ID_IPTV_URI	41	Used in the <code>idType</code> property to indicate an IP broadcast channel identified by a DVB MCAST URI (i.e. <code>dvb-mcast://</code>) or by a URI referencing a HAS or MPEG DASH MPD (i.e. <code>http://</code> or <code>https://</code>), as value for property <code>ipBroadcastID</code> .

7.13.12.3 Properties

This subclause defines the properties of the `Channel` object.

Properties that do not apply in a specific circumstance (e.g. `onid` does not apply unless the channel is of type `ID_DVB_*` or `ID_ISDB_*`) shall be undefined. The mapping to these properties is defined in 8.4.3.

readonly Integer channelType

The type of channel. The value may be indicated by one of the TYPE_* constants defined above. If the type of the channel is unknown then the value shall be "undefined".

NOTE Values of this type between 256 and 511 are reserved for use by related specifications on request by liaison.

readonly Integer idType

The type of identification for the channel, as indicated by one of the ID_* constants defined above

readonly String ccid

Unique identifier of a channel within the scope of the OITF. The ccid is defined by the OITF and shall have prefix 'ccid': e.g. 'ccid:{tunerID}.majorChannel{minorChannel}'.

NOTE The format of this string is platform-dependent.

readonly String tunerID

Optional unique identifier of the tuner within the scope of the OITF that is able to receive the given channel.

readonly Integer onid

DVB or ISDB original network ID.

readonly Integer nid

The DVB or ISDB network ID.

readonly Integer tsid

DVB or ISDB transport stream ID.

readonly Integer sid

DVB or ISDB service ID.

readonly Integer sourceID

ATSC source_ID value.

readonly Integer freq

For analogue channels, the frequency of the video carrier in kHz.

readonly Integer cni

For analogue channels, the VPS/PDC confirmed network identifier.

String name

The name of the channel. Can be used for linking analogue channels without CNI. Typically, it will contain the call sign of the station (e.g. 'HBO').

readonly Integer majorChannel

The major channel number, if assigned. Value undefined otherwise. Typically used for channels of type ID_ATSC_* or for channels of type ID_DVB_* or ID_IPTV_SDS in markets where logical channel numbers are used.

readonly Integer minorChannel

The minor channel number, if assigned. Value undefined otherwise. Typically used for channels of type ID_ATSC_*.

readonly String dsd

For channels of type ID_DVB_SI_DIRECT created through `createChannelObject()`, this property defines the delivery system descriptor (tuning parameters) as defined by DVB-SI 6.2.13.

The `dsd` property provides a string whose characters shall be restricted to the ISO Latin-1 character set. Each character in the `dsd` represents a byte of a delivery system descriptor as defined by DVB-SI EN 300 468 6.2.13, such that a byte at position "i" in the delivery system descriptor is equal the Latin-1 character code of the character at position "i" in the `dsd`.

Described in the syntax of JavaScript: let `sdd[]` be the byte array of a system delivery descriptor, in which `sdd[0]` is the descriptor_tag, then, `dsd` is its equivalent string, if:

`dsd.length==sdd.length` and

for each integer `i`: $0 \leq i < \text{dsd.length}$ holds: `sdd[i] == dsd.charCodeAt(i)`.

readonly Boolean favourite

Flag indicating whether the channel is marked as a favourite channel or not in one of the favourite lists as defined by the property `favIDs`.

readonly StringCollection favIDs

The names of the favourite lists to which this channel belongs (see the `favouriteLists` property on the `ChannelConfig` class).

readonly Boolean locked

Flag indicating whether the current state of the parental control system prevents the channel from being viewed (e.g. a correct parental control pin has not been entered).

Note that this property supports the option of client-based management of parental control without excluding server-side implementation of parental control.

readonly Boolean manualBlock

Flag indicating whether the user has manually blocked viewing of this channel. Manual blocking of a channel will treat the channel as if its parental rating value always exceeded the system threshold.

Note that this property supports the option of client-based management of manual blocking without excluding server-side management of blocked channels.

readonly String ipBroadcastID

If the channel has an `idType` of ID_IPTV_SDS, this property denotes the DVB textual service identifier of the IP broadcast service, specified in the format "ServiceName.DomainName" with the `ServiceName` and `DomainName` as defined in ETSI TS 102 034.

If the Channel has an `idType` of ID_IPTV_URI, this element denotes a URI of the IP broadcast service.

readonly Integer channelMaxBitRate
If the channel has an <code>idType</code> of <code>ID_IPTV_SDS</code> , this property denotes the maximum bitrate associated to the channel.

readonly Integer channelTTR
If the channel has <code>idType</code> <code>ID_IPTV_SDS</code> , this property denotes the <code>TimeToRenegotiate</code> associated to the channel.

readonly Boolean recordable
Flag indicating whether the channel is available to the recording functionality of the OITF. If the value of the <code>pvrSupport</code> property on the <code>application/oipfConfiguration</code> object as defined in 7.3.4.3 is 0, this property shall be <code>false</code> for all <code>Channel</code> objects.

7.13.12.4 Metadata extensions to Channel

7.13.12.4.1 General

This subclause shall apply for OITFs that have indicated `<clientMetadata>` with value "true" and a type attribute with values "bcg", "sd-s", "eit-pf" or "dvb-si" as defined in 9.3.8 in their capability profile.

The OITF shall extend the `Channel` class with the properties and methods described below.

The values of many of these properties may be derived from elements in the BCG metadata. For optional elements that are not present in the metadata, the default value of any property that derives its value from one of those elements shall be `undefined`.

7.13.12.4.2 Properties

String longName
The long name of the channel. If both short and long names are being transmitted, this property shall contain the long name of the station (e.g. 'Home Box Office'). If the long name is not available, this property shall be <code>undefined</code> . The value of this property may be derived from the <code>NAME</code> element that is a child of the BCG <code>ServiceInformation</code> element describing the channel, where the <code>length</code> attribute of the <code>NAME</code> element has the value 'long'.

String description
The description of the channel. If no description is available, this property shall be <code>undefined</code> . The value of this field may be taken from the <code>ServiceDescription</code> element that is a child of the BCG <code>ServiceInformation</code> element describing this channel.

readonly Boolean authorised
Flag indicating whether the receiver is currently authorised to view the channel. This describes the conditional access restrictions that may be imposed on the channel, rather than parental control restrictions.

StringCollection genre
A collection of genres that describe the channel. The value of this field may be taken from the <code>ServiceGenre</code> elements that are children of the BCG <code>ServiceInformation</code> element describing the channel.

Boolean hidden
Flag indicating whether the channel shall be included in the default channel list.

readonly Boolean is3D
Flag indicating whether the channel is a 3D channel.

readonly Boolean isHD
Flag indicating whether the channel is an HD channel.

string logoURL
The URL for the default logo image for this channel. The value of this field may be derived from the value of the first Logo element that is a child of the BCG ServiceInformation element describing the channel. If this element specifies anything other than the URL of an image, the value of this field shall be undefined .

7.13.12.4.3 Methods

String getField(String fieldId)		
Description	Get the value of the field referred to by fieldId that is contained in the BCG metadata for this channel. If the field does not exist, this method shall return undefined .	
Arguments	fieldId	The name of the field whose value shall be retrieved.

String getLogo(Integer width, Integer height)		
Description	Get the URI for the logo image for this channel. The width and height parameters specify the desired width and height of the image; if an image of that size is not available, the URI of the logo with the closest available size not exceeding the specified dimensions shall be returned. If no image matches these criteria, this method shall return null . The URI returned shall be suitable for use as the SRC attribute in an HTML IMG element or as a background image. The URIs returned by this method will be derived from the values of the Logo elements that are children of the BCG ServiceInformation element describing the channel.	
Arguments	width	The desired width of the image.
	height	The desired height of the image.

7.13.13 The FavouriteListCollection class

7.13.13.1 General

```
typedef Collection<FavouriteList> FavouriteListCollection
```

The **FavouriteListCollection** class represents a collection of **FavouriteList** objects. See Annex I for the definition of the collection template. In addition to the methods and properties defined for generic collections, the **FavouriteListCollection** class supports the additional methods defined below.

7.13.13.2 Methods

FavouriteList getFavouriteList(String favID)		
Description	Return the first favourite list in the collection with the given favListID .	
Arguments	favID	The ID of a favourite list.

7.13.13.3 Extensions to FavouriteListCollection

If an OITF has indicated support for extended tuner control (i.e. by giving value `true` to element `<extendedAVControl>` as specified in 9.3.7 in its capability description), the OITF shall support the following additional constants and methods on the `FavouriteListCollection` object.

The functionality as described in this subclause is subject to the security model of 10.1.4.9.

FavouriteList createFavouriteList(String name)		
Description	Create a new favourite list and add it to the collection. The new favourite list shall be returned.	
Arguments	name	The name to be associated to the new favourite list.

Boolean remove(Integer index)		
Description	Remove the list at the specified index from the collection. This method shall return <code>true</code> if the operation succeeded, or <code>false</code> if an invalid index was specified.	
Arguments	index	The index of the list to be removed.

Boolean commit()	
Description	Commit any changes to the collection to persistent storage. This method shall return <code>true</code> if the operation succeeded, or <code>false</code> if it failed (e.g. due to insufficient space to store the collection). If a server has indicated that it requires control of the tuner functionality of an OITF in the server capability description for a particular service, then the OITF should send an updated client channel listing to the server using HTTP POST over TLS as described in 4.9.2.2.

Boolean activateFavouriteList(String favID)	
Description	Active the favourite list from the collection. This method shall return <code>true</code> if the operation succeeded, or <code>false</code> if an invalid index was specified. A newly created favourite list has to be committed before it can be activated.
Arguments	favID

7.13.14 The FavouriteList class

7.13.14.1 General

```
typedef Collection<Channel> FavouriteList
```

The `FavouriteList` class represents a list of favourite channels. See Annex I for the definition of the collection template. In addition to the methods and properties defined for generic collections, the `FavouriteList` class supports the additional properties and methods defined below.

In order to preserve backwards compatibility with already existing DAE content the JavaScript `toString()` method shall return the `FavouriteList.id` for `FavouriteList` objects.

7.13.14.2 Properties

readonly String favID
A unique identifier by which the favourite list can be identified.

String name
A descriptive name given to the favourite list.

7.13.14.3 Methods

Channel getChannel(String channelId)		
Description	Return the first channel in the favourite list with the specified channel identifier. Returns null if no corresponding channel can be found.	
Arguments	channelID	The channel identifier of the channel to be retrieved, which is a value as defined for property ccid of the Channel object or a value as defined for property ipBroadcastID of the Channel object as defined in 7.13.12.

Channel getChannelByTriplet(Integer onid, Integer tsid, Integer sid)		
Description	Return the first (IPTV or non-IPTV) channel in the list that matches the specified DVB or ISDB triplet (original network ID, transport stream ID, service ID). Where no channels of type ID_ISDB_* or ID_DVB_* are available, or no channel identified by this triplet are found, this method shall return null .	
Arguments	onid	The original network ID of the channel to be retrieved.
	tsid	The transport stream ID of the channel to be retrieved. If set to null the client shall retrieve the channel defined by the combination of onid and sid . This makes it possible to retrieve the correct channel also in case a remultiplexing took place which led to a changed tsid .
	sid	The service ID of the channel to be retrieved.

Channel getChannelBySourceID(Integer sourceID)		
Description	Return the first (IPTV or non-IPTV) channel in the list with the specified ATSC source ID. Where no channels of type ID_ATSC_* are available, or no channel with the specified source ID is found in the channel list, this method shall return null .	
Arguments	sourceID	The ATSC source_ID of the channel to be returned.

7.13.14.4 Extensions to FavouriteList

If an OITF has indicated support for extended tuner control (i.e. by giving value **true** to element **<extendedAVControl>** as specified in 9.3.7 in its capability description), the OITF shall support the following additional constants and methods on the **FavouriteList** object.

When the **FavouriteList** object is updated with new or removed channels it does not take effect until the object is committed. Only after **commit()** will the updates of a **FavouriteList** object become available to other DAE applications.

The name property of the **FavouriteList** object shall be read/write for OITFs which are controlled by a service provider. The following methods shall also be supported:

Boolean insertBefore(Integer index, String ccid)		
Description	Insert a new favourite into the favourites list at the specified index. In order to add a ccid at the end of the favourite list the index shall be equal to length . This method shall return true if the operation succeeded, or false if an invalid index was specified (e.g. index > (length)).	
Arguments	index	The index in the list before which the favourite should be inserted.
	ccid	The ccid of the channel to be added.

Boolean remove (Integer index)		
Description	Remove the item at the specified index from the favourites list. Returns true if the operation succeeded, or false if an invalid index was specified.	
Arguments	index	The index of the item to be removed.

Boolean commit ()	
Description	Commit any changes to the favourites list to persistent storage. This method shall return true if the operation succeeded, or false if it failed (e.g. due to insufficient space to store the list on the OITF). If a server has indicated that it requires control of the tuner functionality of an OITF in the server capability description for a particular service, then the OITF should send an updated client channel listing to the server using HTTP POST over TLS as described in 4.9.2.2.

7.13.15 Extensions to video/broadcast for channel scan

The subclause has been merged with the ChannelConfig class (see 7.13.10 above).

7.13.16 The ChannelScanEvent class

The contents of this subclause have been merged with 7.13.10.4 above.

7.13.17 The ChannelScanOptions class

7.13.17.1 General

The ChannelScanOptions class defines the options that should be applied during a channel scan operation. This class does not define parameters for the channel scan itself.

7.13.17.2 Properties

Integer channelType
The types of channel that should be discovered during the scan. Valid values are TYPE_RADIO, TYPE_TV, or TYPE_OTHER or TYPE_ALL as defined in 7.13.12.2.

Boolean replaceExisting
If true , any existing channels in the channel list managed by the OITF shall be removed and the new channel list shall consist only of channels found during the channel scan operation. If false , any channels discovered during the channel scan shall be added to the existing channel list.

7.13.18 The ChannelScanParameters class

This is an empty class that acts as the base interface for channel scan parameters specific to certain types of broadcast network.

7.13.19 The DVBTChannelScanParameters class

The DVBTChannelScanParameters class represents the parameters needed to perform a channel scan on a DVB-T or DVB-T2 network. This class implements the interface defined by ChannelScanParameters, with the following additions.

The properties that are undefined when performing startScan() are considered to be auto detected.

Integer startFrequency
The start frequency of the scan, in kHz.

Integer endFrequency
The end frequency of the scan, in kHz.

Integer raster
The raster size represented in kHz (typically 7 000 or 8 000).

String ofdm

The Orthogonal Frequency Division Multiplexing (OFDM) for the indicating frequency. Valid values are:

Value	Description
MODE_1K	OFDM mode 1K
MODE_2K	OFDM mode 2K
MODE_4K	OFDM mode 4K
MODE_8K	OFDM mode 8K
MODE_16K	OFDM mode 16K
MODE_32K	OFDM mode 32K

Integer modulationModes

The modulation modes to be scanned. Valid values are:

Value	Description
1	QPSK modulation
4	QAM16 modulation
8	QAM32 modulation
16	QAM64 modulation
32	QAM128 modulation
64	QAM256 modulation

More than one of these values may be ORed together in order to indicate that more than one modulation mode should be scanned.

String bandwidth

The expected bandwidth. Valid values are:

Value	Description
BAND_1.7MHZ	1.7 MHz bandwidth
BAND_5MHZ	5 MHz bandwidth
BAND_6MHZ	6 MHz bandwidth
BAND_7MHZ	7 MHz bandwidth
BAND_8MHZ	8 MHz bandwidth
BAND_10MHZ	10 MHz bandwidth

7.13.20 The DVBSChannelScanParameters class

7.13.20.1 General

The DVBSChannelScanParameters class represents the parameters needed to perform a channel scan on a DVB-S or DVB-S2 network. This class implements the interface defined by ChannelScanParameters, with the following additions.

The properties that are undefined when performing startScan() are considered to be auto detected.

7.13.20.2 Properties

Integer startFrequency
The start frequency of the scan, in kHz.

Integer endFrequency
The end frequency of the scan, in kHz.

Integer modulationModes

The modulation modes to be scanned. Valid values are:

Value	Description
1	QPSK modulation
2	8PSK modulation
4	QAM16 modulation

More than one of these values may be ORed together in order to indicate that more than one modulation mode should be scanned.

String symbolRate
A comma-separated list of the symbol rates to be scanned, in symbols/s.

Integer polarisation											
The polarisation to be scanned. Valid values are:											
	<table><tr><th>Value</th><th>Description</th></tr><tr><td>1</td><td>Horizontal polarisation</td></tr><tr><td>2</td><td>Vertical polarisation</td></tr><tr><td>4</td><td>Right-handed/clockwise circular polarisation</td></tr><tr><td>8</td><td>Left-handed/counter-clockwise circular polarization</td></tr></table>	Value	Description	1	Horizontal polarisation	2	Vertical polarisation	4	Right-handed/clockwise circular polarisation	8	Left-handed/counter-clockwise circular polarization
Value	Description										
1	Horizontal polarisation										
2	Vertical polarisation										
4	Right-handed/clockwise circular polarisation										
8	Left-handed/counter-clockwise circular polarization										
More than one of these values may be ORed together in order to indicate that more than one polarisation should be scanned.											

String codeRate
The code rate, e.g. "3/4" or "5/6".

Number orbitalPosition
The orbitalPosition property is used to resolve DiSEqC switch/motor. The value is the orbital position of the satellite, negative value for west, positive value for east. For example, Astra 19,2 East would have orbitalPosition 19,2. Thor 0,8 West would have orbitalPosition -0,8.

Integer networkId
The network ID of the network to be scanned, or undefined if all networks should be scanned.

7.13.21 The DVBCChannelScanParameters class

7.13.21.1 General

The DVBCChannelScanParameters class represents the parameters needed to perform a channel scan on a DVB-C or DVB-C2 network. This class implements the interface defined by ChannelScanParameters, with the following additions.

The properties that are undefined when performing startScan() are considered to be auto detected.

7.13.21.2 Properties

Integer startFrequency
The start frequency of the scan, in kHz.

Integer endFrequency
The end frequency of the scan, in kHz.

Integer raster
The raster size represented in kHz (typically 7 000 or 8 000).

Boolean startNetworkScanOnNIT
The scan mode for scanning. A false value indicates to scan complete range, a true value indicates scan terminates when a valid NIT is found. The frequency scan is replaced by a scan based on NIT. If networkId is set and the value of this property is set to true , the scan continues until there is a match on both.

Integer modulationModes

The modulation modes to be scanned. Valid values are:

Value	Description
4	QAM16 modulation
8	QAM32 modulation
16	QAM64 modulation
32	QAM128 modulation
64	QAM256 modulation
128	QAM1024 modulation
256	QAM4096 modulation

More than one of these values may be ORed together in order to indicate that more than one modulation mode should be scanned.

String symbolRate
A comma-separated list of the symbol rates to be scanned, in symbols/s.

Integer networkId
The network ID of the network to be scanned, or undefined if all networks should be scanned.

7.13.22 Extensions to video/broadcast for synchronization

7.13.22.1 General

The OITF shall support the following additional methods on the video/broadcast object, in order to enable synchronization to broadcast events.

void addStreamEventListener(String targetURL, String eventName, EventListener listener)		
Description	Add a listener for the specified DSM-CC stream event. Event triggers are carried in the stream as MPEG private data subclauses. For robustness, the subclause describing a particular trigger may be repeated several times. Each subclause has a version number which is used to disambiguate a new trigger for the same event (which will have a different version number) from a repeated instance of a previous trigger (which will have the same version number). When OITF detects a trigger corresponding to an event for which a listener has been registered, a DOM StreamEvent shall be dispatched. An event shall also be dispatched in case of error. An OITF shall dispatch only one DOM StreamEvent per unique trigger detected. Repeated instances of the same trigger shall not cause a new DOM StreamEvent to be dispatched. A new trigger for the same event (i.e. an MPEG private data subclause for the same event but with an updated version number) shall cause a new DOM StreamEvent to be dispatched.	
Arguments	targetURL	The URL of the DSM-CC StreamEvent object or the event description file describing the event as defined in 8.2 of ETSI TS 102 809.
	eventName	The name of the event (in the DSM-CC StreamEvent object) that should be subscribed to.
	listener	The listener for the event.

void removeStreamEventListener(String targetURL, String eventName, EventListener listener)		
Description	Remove a stream event listener for the specified stream event name.	
Arguments	targetURL	The URL of the DSM-CC StreamEvent object or the event description file describing the event as defined in 8.2 of ETSI TS 102 809.
	eventName	The name of the event (in the DSM-CC StreamEvent object) whose subscription should be removed.
	listener	The listener for the event.

7.13.22.2 The StreamEvent class

The **StreamEvent** class is a subclass of the DOM Event class which notifies an application that a synchronisation trigger in a broadcast stream has been detected. This event also notifies an application when the event is no longer being monitored.

Instances of this event are directly dispatched to the event target, and will not bubble nor capture.

readonly String eventName
The name of the stream event.

readonly String data
Data of the DSM-CC StreamEvent's event encoded in hexadecimal. For EXAMPLE: "0A10B81033" (for a message 5 bytes long).

readonly String text
Text data of the DSM-CC StreamEvent's event as a string, assuming UTF-8 as the encoding for the DSM-CC StreamEvent's event. Characters that cannot be transcoded shall be skipped.

readonly String status
The status of the event. Equal to "trigger" when the event is dispatched in response to a trigger in the stream or "error" when an error occurred (e.g. attempting to add a listener for an event that does not exist, or when a StreamEvent object with registered listeners is removed from the carousel). An event shall be dispatched with an error status if: - the StreamEvent object pointed to by targetURL is not found in the carousel or via broadband; - the StreamEvent object pointed to by targetURL does not contain the event specified by the eventName parameter; - the carousel containing the event cannot be mounted; - the elementary stream which contains the StreamEvent event descriptor is no longer being monitored (e.g. due to another monitoring request or because it disappears from the PMT); - the event description file pointed to by targetURL is not available or does not have the correct syntax. Once an error is dispatched, the listener shall be automatically unregistered by the OITF.

7.13.23 The ATSCChannelScanParameters class

7.13.23.1 General

The ATSCChannelScanParameters class represents the parameters needed to perform a channel scan on an ATSC-T network. This class implements the interface defined by ChannelScanParameters, with the following additions.

The properties that are undefined when performing startScan() are considered to be auto detected.

7.13.23.2 Properties

Integer startFrequency
The start frequency of the scan, in kHz.

Integer endFrequency
The end frequency of the scan, in kHz.

Integer raster
The raster size represented in kHz, typically 6 000 as this is the ATSC channel separation.

Integer modulationModes	
The modulation modes to be scanned. Valid values are:	
Value	Description
2	2VSB
4	4VSB
8	8VSB
16	16VSB
More than one of these values may be arithmetically summed in order to indicate that more than one modulation mode should be scanned.	

7.14 Media playback APIs

7.14.1 General

Subclause 7.14 specifies several extensions to the audio object and the video object defined in 5.7.1 of CEA-2014-A. Subclause 7.14.11 defines the audio playback from memory API.

7.14.2 The A/V Control object

7.14.2.1 General

The HTML object element (defined in 4.8.4 of the HTML5 specification as referenced in IEC 62766-5-2) shall support presentation of video and audio or just audio. This presentation shall be indicated by setting the type attribute to one of the "video/" or "audio/" MIME types that is listed in Clause 4 of IEC 62766-2-1:2016 and corresponds to a combination of system layer format, video format and audio format that is supported by the OITF.

Table 9 lists the properties of the A/V control object with a type attribute of either video or audio.

Table 10 lists additional properties of the A/V control object with a type attribute of video.

Table 11 lists the methods of the A/V control object with a type attribute of either video or audio.

Table 12 lists the additional methods of the A/V control object with a type attribute of video.

When an HTML object is presenting video or audio as defined in this subclause, the following requirements shall apply:

- HTML objects presenting video and/or audio shall be accessible through the DOM and shall support the properties and methods defined by the tables below in addition to those of `HTMLObjectElement` as defined in the HTML5 specification as referenced in IEC 62766-5-2.

Table 9 – Properties of the A/V Control object when the type attribute refers to video or audio

Property type and name	Property definition
String data	String data [RW] – media URL. If the value of data is changed while media is playing, playback is stopped (resulting in a play state change). The default value is the empty string. If the value of this attribute is changed, the related data-attribute inside the DOM tree should be changed accordingly. If the value of this attribute is set to an empty string or is changed, the resources (files, server connections, etc...) currently owned by the object shall be released. The value set in this property may include a temporal fragment interval according to 4.2.1 of W3C Media Fragments URI 1.0, in which case the derived begin time and end time shall serve as bounds for playback. The Normal Play Time format shall be used. The begin time shall behave as start-of-media and the end time shall behave as end-of-media. If the value of temporal fragment interval is changed then there will be no change in the play state unless the interval is changed to period outside of the current play position.
readonly Number playPosition	The play position in number of milliseconds since the beginning as denoted by the server (i.e. in relation to NPT 0.0 as described in 3.6 of IETF RFC 2326) of the media referenced by attribute data when data refers to a single media item. If the play position cannot be determined, the playPosition shall be undefined.
readonly Number playTime	The estimated total duration in milliseconds of the media referenced by data when data refers to a single media item. If the duration of the media cannot be determined, the playTime shall be undefined.
readonly Number playState	Indication of the current play state as follows. 0 – Stopped; user (or script) has stopped playback of the current media, or playback has not yet started. 1 – Playing; the current media pointed to by data is currently playing. 2 – Paused; the current media pointed to by data has been paused. 3 – Connecting; connect to media server, i.e. waiting for connection to media server to be established, upon first connection or after the connection was lost. In addition, DRM rights necessary for playback of protected content are also retrieved during this state. 4 – Buffering; the buffer is being filled in order to have sufficient data available to initiate or continue playback. In this state, playback is stalled due to insufficient data in the buffer to continue playback. The player waits until sufficient data has been buffered to continue playback. For video objects, whilst being in this state, the player should show the last completed video frame that was shown before entering this state. This playstate is an intermediate state to reach playState 1 ('playing'). The OITF should buffer the content in the background whilst in playState 2 ('paused'). However, this background buffering does not result into a state change to state 4. 5 – Finished; the playback of the current media has reached the end of the media. 6 – Error; an error occurred during media playback, preventing the current media to start/continue playing.
readonly Number error	Error details; only significant if the value of playState equals 6: 0 – A/V format not supported. 1 – cannot connect to server or connection lost. 2 – unidentified error. 3 – insufficient resources. 4 – content corrupt or invalid. 5 – content not available. 6 – content not available at given position.

Property type and name	Property definition
readonly Number speed	Play speed relative to real-time as defined by 5.7.1.f.6 of CEA-2014-A.
Object onPlayStateChange	DOM-0 event handler called when the value of the playState property changes as defined by 5.7.1.f.9 of CEA-2014-A.

Table 10 – Additional properties of the A/V Control object when the type attribute refers to video

Property type and name	Property definition
String width	The width of the area used for rendering the video object. This property is only writable if property fullScreen has value false . The effect of changes to width shall be in accordance with requirement 5.7.1.c of CEA-2014-A.
String height	The height of the area used for rendering the video object. This property is only writable if property fullScreen has value false . The effect of changes to height shall be in accordance with requirement 5.7.1.c of CEA-2014-A.
readonly Boolean fullScreen	Indicates whether an object presenting video is in full screen mode or not – as defined in by 5.7.1.g.3 of CEA-2014-A.
Object onFullScreenChange	DOM-0 event handler called when the value of fullScreen changes as defined in by 5.7.1.g.4 of CEA-2014-A.
Object onfocus	DOM-0 event handler called when the object gains focus as defined by 5.7.1.g.7 of CEA-2014-A.
Object onblur	DOM-0 event handler called when the object loses focus as defined by 5.7.1.g.8 of CEA-2014-A.

IECNORM.COM : Click to view the full PDF of IEC 62766-5-1:2017

Table 11 – Methods of the A/V Control object when the type attribute refers to video or audio

Method signature	Method definition
Boolean play(Number speed)	Plays the media referenced by data, starting at the current play position denoted by playPosition , at the supported speed closest to the value of attribute speed. Negative speeds reverse playback. If no speed is specified, it defaults to 1. A speed of 0 will pause playback. This method shall always return true . If the playback reached the beginning of the media at rewind playback speed, then the play state shall be changed to 2 ('paused'). A play speed event (see 7.14.4.2) shall be generated when the operation has completed, regardless of the new play speed. If the play speed is not changed, the argument of the event shall be set to the previous play speed.
Boolean stop()	Stops playback and resets the playPosition to 0 as defined by 5.7.1.f.12 of CEA-2014-A.
Boolean seek(Number pos)	If seek() is called while the player is in state 1 ("playing"), then it sets the current play position (in milliseconds) to the value of pos and may change play state to 4 ('buffering'). If the player is in state 2 ('paused'), then the seek() method seeks to the new position, but the play state and the rendered image is not changed. If the player is in states 0 ("stopped"), 5 ("finished") or 6 ("error"), then the new play position shall be retained and shall be used (if possible) as the starting position for playing back the content item indicated by the data property when the play() method is called. NOTE Changing the content item resets the play position to the beginning of the new content item. If the player is in states 3 ("connecting") or 4 ("buffering") then the seek() method seeks to the new play position and may change play state to 3 ("connecting"). If the new playback position is valid, the value of the playPosition attribute shall be set to the new value before this method returns. Returns true if the method succeeded, and false otherwise. A play position event (see 7.14.4.2) will be generated when the operation has completed, regardless of the success of the operation. If the operation fails, the argument of the event shall be set to the previous play position.
Boolean setVolume(Number volume)	Sets the audio volume as defined by 5.7.1.f.14 of CEA-2014-A.

Table 12 – Additional methods of the A/V Control object when the type attribute refers to video

Method signature	Method definition
setFullScreen(Boolean fullscreen)	Sets the object to full screen mode or windowed mode as defined by 5.7.1.g.5 of CEA-2014-A.
focus()	Sets the input focus to this object as defined by 5.7.1.g.6 of CEA-2014-A.

- b) In addition to the properties and methods listed above, the Table 13 lists other requirements from CEA-2014-A that shall also apply for the A/V Control object as defined in this subclause.

Table 13 – Additional applicable requirements from CEA-2014

CEA-2014 requirement	Summary
5.7.1.a.3	Indication of the initial aspect ratio of the video content.
5.7.1.b.3	Access to both video and audio objects by name/id through the HTMLDocument interface.
5.7.1.c	Full-screen or windowed mode for video objects.
5.7.1.d	CSS properties that apply to video objects.
5.7.1.e	CSS <code>z-index</code> , <code>opacity</code> and RGBA colour values. NOTE In this standard, the value of the "<overlay>" element is never "none" and hence <code>z-index</code> and <code>opacity</code> are required to be supported.

7.14.2.2 State diagram for A/V Control objects

The state transition diagram in Figure 16 should be used for an A/V Control object.

IECNORM.COM : Click to view the full PDF of IEC 62766-5-1:2017

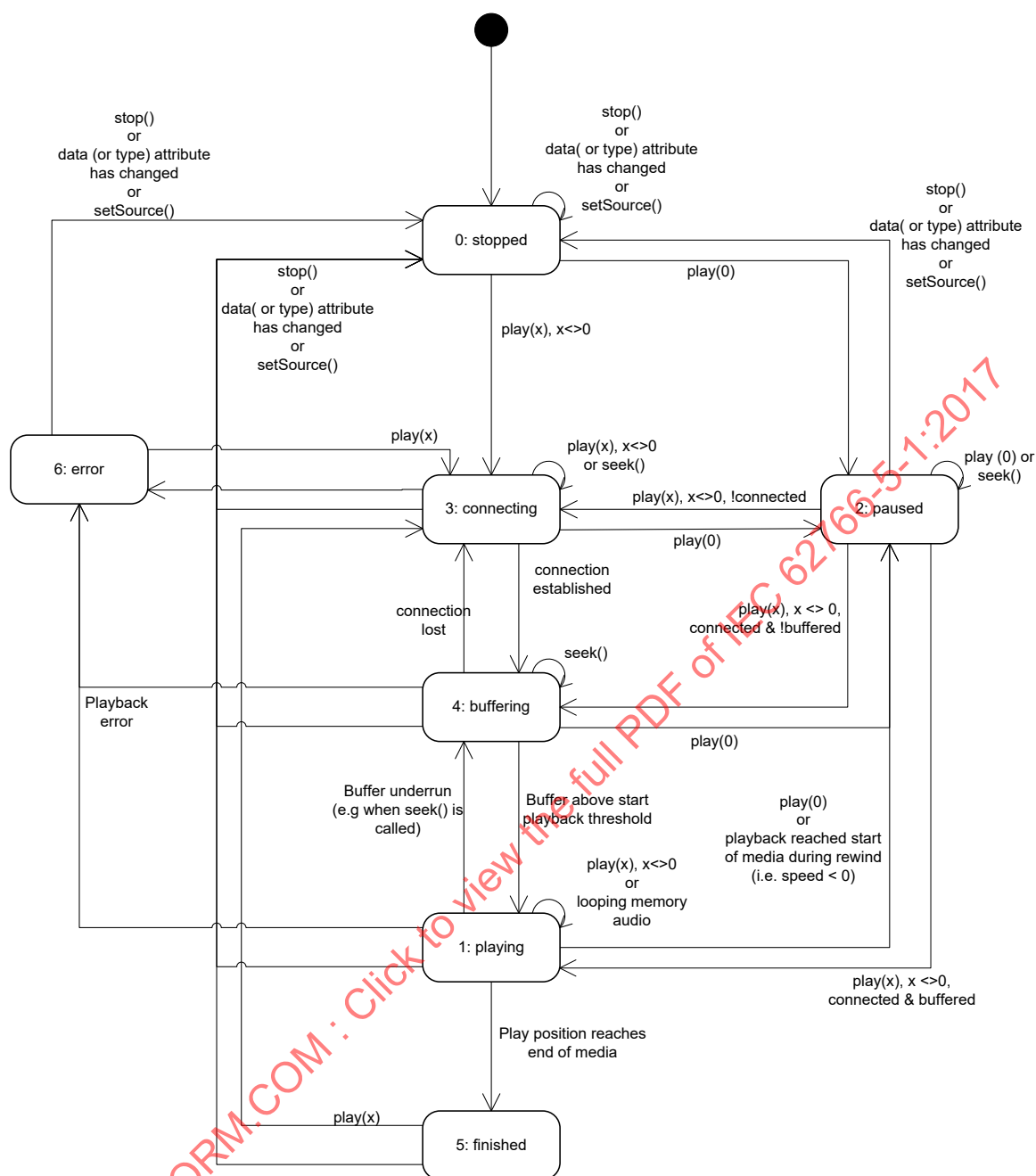


Figure 16 – State diagram for embedded A/V Control objects (normative)

The following clarifications apply.

- A detailed description for all the states in this state diagram is given above in the definition of the `playState` property.
- If the value of the `allocationMethod` property is `DYNAMIC_ALLOCATION`, the following shall apply:
 - scarce resources for playback using the A/V Control object, such as the MPEG decoder, are claimed during state 3 ('connecting'), state 4 ('buffering') or during state transitions from state 3 ('connecting') to state 4 ('buffering'), from state 4 ('buffering') to state 1 ('playing') or from state 0 ('stopped') or from state 3 ('connecting') to state 2 ('paused').
 - If at any point in time during playback, the scarce resources are not available anymore due to a resource conflict, then the play state of the A/V Control object shall be set to 6 ('error') with a detailed error code of 3 ('insufficient resources').

- Scarce resources for playback using the A/V Control object shall be released when state 6 ('error') or 0 ('stopped') or 5 ('finished') are reached.
- c) In addition, if the A/V Control object gets destroyed, e.g. because another URL is loaded into the containing window, scarce resources claimed for playback using the A/V Control object shall be released, except in cases described for the optional persist property of A/V Control objects.
- d) When the data attribute and/or the type attribute of the `HTMLObjectElement` representing the A/V Control object is given a different value, the object shall go to state 0 ('stopped').
- e) For playback of DRM protected content, the rights for playback are retrieved during state 3 ('connecting').
- f) If the play position reaches the end of the available content the A/V Control object shall be set to state 5 ('finished') in addition to generating a playback speed change of zero.
- g) If there is an attempt to `play()` with a speed in the positive direction (forward or > 0) and there is no content available then the request fails.
- h) If the play position reaches the beginning of the available content the A/V Control object shall be set to state 2 ('paused') in addition to generating a playback speed change of zero.
- i) If there is an attempt to `play()` with a speed in the negative direction (rewind or < 0) and there is no content available then the request fails.
- j) If `seek()` is performed beyond the available content the request is rejected and the current playout is maintained.
- k) When a A/V Control object stops being rendered as defined in 10.1 of the HTML5 specification as referenced by IEC 62766-5-2, an OITF may release scarce resources allocated for that object. Vice versa, an A/V Control object which is not visible but is still being rendered shall still be decoding video if it is in the playing state and any audio associated with the currently playing media will still be audible. State transitions caused by calls to methods on the A/V Control object, or due to permanent or transient errors, will occur as shown above regardless of the visibility of the object. Subclause 4.5.5 describes the effect on scarce resources when an A/V Control object is removed from the DOM tree.

NOTE As implied by the text above, rendering state and visibility are not equivalent. This implies, just to give two examples, that `display:none` will affect the object state while `visibility:hidden` will not. When an A/V Control object is destroyed (e.g. by the A/V Control object being garbage collected, or because of a page transition within the application), presentation of streamed audio or video is terminated.

- l) When not presenting video, the A/V Control object shall be rendered as an opaque black rectangle.

7.14.2.3 Using an A/V Control object to play streaming content

If an A/V Control object is used to play streamed content using either RTSP or HTTP the OITF then the following holds:

- a) If `play(0)` is called in state 0 ('stopped'), the A/V Control object shall automatically go to play state 2 ('paused'). The necessary resources are secured and no external signalling is performed.
- b) If `play(0)` is called in the connecting or buffering state, the A/V Control object shall automatically go to play state 2 ('paused').
- c) If `play()` is called in the paused state with an argument other than 0, the A/V Control object shall transition to one of the following states as follows:
 - if there is no connection to the server, the A/V Control object shall transition to the connecting state;
 - if there is a connection to the server but insufficient content is buffered to resume playback immediately, the A/V Control object shall transition to the buffering state;
 - if there is a connection to the server and sufficient content is buffered to resume playback immediately, the A/V Control object shall transition to the playing state.

7.14.2.4 Using an A/V Control object to play downloaded content

If an A/V Control object is used to play content that has been downloaded and stored on the OITF (by using method `setSource()` as defined in 7.14.8) then the following holds:

- a) If the download was triggered using `registerDownloadURL` or the download was triggered using a content access download descriptor with `<TransferType>` value "playable_download" as defined in Clause C.1, then:
 - If the `play()` method is called before sufficient data has been download to initiate playback, then the play state of the A/V Control object shall be set to 6 ('error') with a detailed error code of 5 ("content not available").
- b) If the downloaded content was triggered using a content access download descriptor with `<TransferType>` value "full_download" as defined in Clause C.1, then:
 - If the `play()` method is called whilst the content is still downloading and has not yet successfully completed, then the play state of the A/V Control object shall be set to 6 ('error') with a detailed error code of 5 ("content not available").

7.14.2.5 Using an A/V Control object to play recorded content

If an A/V Control object is used to play content that has been recorded or is being recorded on the OITF (by using method `setSource()` as defined in 7.14.8) then the following holds:

- If the `play()` method is called before sufficient data has been recorded to initiate playback, then the play state of the A/V Control object shall be set to 6 ('error') with a detailed error code of 5 ("content not available").

7.14.2.6 Using the A/V Control object to play content fragments

If the OITF indicates support through the `temporalClipping` capability indicator (see 9.3.23) for playing content fragments then it shall support the Media Fragment URI W3C Media Fragments URI 1.0 according to 8.3.2.

7.14.2.7 User Input and the A/V Control object

If an A/V Control object has input focus:

- The OITF shall not block execution of scripts of the document from which the focus was moved to the video object, even when the video is playing full-screen and has input focus.
- The OITF shall not handle the `VK_OK`, `VK_PLAY`, `VK_PAUSE`, `VK_PLAY_PAUSE`, `VK_STOP`, `VK_FAST_FWD`, `VK_REWIND`, `VK_NEXT` or `VK_PREV` keys.

7.14.3 Extensions to A/V Control object for playback through Content-Access Streaming Descriptor

As specified in 4.8.2, an OITF shall support setting up the A/V stream using the information provided by a valid Content Access Streaming Descriptor referred to by the 'data' attribute. To this end, the OITF shall fetch the Content Access Streaming Descriptor from the URL provided by the "data" attribute, after which the descriptor shall be interpreted, resulting in an appropriate `<ContentURL>` to be selected (e.g. based on which DRM system the OITF supports). The OITF shall then initiate a streaming CoD session to the selected `<ContentURL>`, after which playback can be started when the `play()` method is invoked.

The OITF shall pass included DRM-information of the selected content and DRM system ID as part of the `<DRMControlInformation>` elements of a Content Access Streaming Descriptor to the DRM agent, if it supports a DRM agent with a matching `DRMSystemID` as per 9.3.11.

If the Content Access Streaming Descriptor is not valid according to the XML Schema and semantics as defined in Annex C.2, the A/V Control object shall go to playState 6 (i.e. error),

with error value 4, which is defined as follows in addition to the error states identified by bullet 5 of [Req. 5.7.1.f] of CEA-2014-A:

4: content corrupt or invalid.

For more information about setting up the A/V stream based on a Content Access Streaming descriptor, see 4.8.2, Clause 8, and CEA-2014-A.

7.14.4 Extensions to A/V Control object for trickmodes

7.14.4.1 Properties

The following additional properties shall be supported on the audio object and video object defined in subclause 5.7.1 of CEA-2014-A.

function **onPlaySpeedChanged**(Number speed)

The function that is called when the playback speed of the media changes.

The specified function is called with one argument, **speed**, which is defined as follows:

- **Number speed** – the playback speed of the media at the time the event was dispatched.

The behaviour of the A/V Control object when the end of media (or the end of the currently-available media) is reached is defined in 7.14.2.

function **onPlayPositionChanged**(Integer position)

The function that is called when change occurs in the play position of the media due to the use of trick play functions.

The specified function is called with one argument, **position**, which is defined as follows:

- **Integer position** – the playback position of the media at the time the event was dispatched, measured in milliseconds since the beginning of the referenced media as denoted by the server.

The behaviour of the A/V Control object when the end of media (or the end of the currently-available media) is reached is defined in 7.14.2.

readonly Number **playSpeeds**[]

Returns an ordered list of playback speeds, expressed as values relative to the normal playback speed (1,0), at which the currently specified A/V content can be played (either through an CEA-2014 audio or video object), or **undefined** if the supported playback speeds are not (yet) known.

function **onPlaySpeedsArrayChanged**()

The function that is called when the **playSpeeds** array values have changed. An application that makes use of the **playSpeeds** array needs to read the values of the **playSpeeds** property again.

readonly String **oifSourceIPAddress**

The OITF source IP address for RTSP or HTTP signalling, as well as, the address where the RTSP stream is expected to arrive. The information shall be available in "buffering", "paused" or "playing" states.

readonly String **oifSourcePortAddress**

The OITF Port Address where the RTSP stream is expected to arrive. The information shall be available in "buffering", "paused" or "playing" states.

Boolean **oifNoRTSPSessionControl**

When the **oifNoRTSPSessionControl** is set to **true**, then the OITF shall not signal the RTSP messages DESCRIBE, SETUP or TEARDOWN.

String oItfRTSPSessionId
The sessionId to be used by the A/V Control object when signalling RTSP. This property is only applicable when property oItfNORTSPSessionControl is set to true .

7.14.4.2 Events

For the intrinsic events listed in the table below, a corresponding DOM event shall be generated in the following manner.

Intrinsic event	Corresponding DOM event	DOM Event properties
onPlaySpeedChanged	PlaySpeedChanged	Bubbles: No Cancellable: No Context Info: speed
onPlayPositionChanged	PlayPositionChanged	Bubbles: No Cancellable: No Context Info: position
onPlaySpeedsArrayChanged	PlaySpeedsArrayChanged	Bubbles: No Cancellable: No Context Info: None

The DOM events are directly dispatched to the event target, and will not bubble nor capture. Applications should not rely on receiving these events during the bubbling or the capturing phase. Applications that use DOM event handlers shall call the `addEventListener()` method on the A/V Control object itself. The third parameter of `addEventListener`, i.e. "useCapture", will be ignored.

7.14.5 Extensions to A/V Control object for playback of selected components

To support the selection of specific A/V components for playback (e.g. a specific subtitle language, audio language, or camera angle), the classes defined in 7.16.5.3 to 7.16.5.6 shall be supported and the constants, properties and methods defined in 7.16.5.2 shall be supported on the A/V Control object.

7.14.6 Extensions to A/V Control object for parental rating errors

7.14.6.1 General

For parental rating errors during playback of A/V content through the CEA-2014 A/V Control object (as defined in subclause 5.7.1 of CEA-2014-A), an OITF shall support the following intrinsic event properties and corresponding DOM events for the CEA-2014 A/V Control object.

function	onParentalRatingChange(	String
	contentID, ParentalRatingCollection ratings, String DRMSystemID, Boolean blocked)	
The function that is called whenever the parental rating of the content being played inside the A/V Control object changes. These events may occur at the start of a new content item, or during playback of a content item (e.g. during playback of A/V streaming content). The specified function is called with four arguments contentID, ratings, DRMSystemID, and blocked which are defined as follows: • String contentID – the content ID to which the parental rating change applies. If the event is generated by the DRM system, it shall be the unique identifier for that content in the context of the DRM system (i.e. in the case of Marlin BB it is the Marlin contentID, in the case of CSPG-CI+ the value of this field is null). Otherwise, it may be null or undefined. • ParentalRatingCollection ratings – the parental ratings of the currently playing content. The ParentalRatingCollection object is defined in 7.9. • String DRMSystemID – the DRM System ID of the DRM system that generated the event as defined by element DRMSystemID in 4.4.3 of IEC 62766-3:2016. The value shall be null if the parental control is not enforced by a particular DRM system. • Boolean blocked – flag indicating whether consumption of the content is blocked by the parental control system as a result of the new parental rating value.		

function	onParentalRatingError(	String	contentID,
	ParentalRatingCollection ratings, String DRMSystemID)		
The function that is called when a parental rating error occurs during playback of A/V content inside the A/V Control object, and is triggered whenever one or more parental ratings are discovered and none of them are valid. A valid parental rating is defined as one which uses a parental rating scheme that is supported by the OITF and which has a parental rating value that is supported by the OITF. The specified function is called with three arguments contentID, rating, and DRMSystemID which are defined as follows: • String contentID – the content ID to which the parental rating error applies. If the event is generated by the DRM system, it shall be the unique identifier for that content in the context of the DRM system (i.e. in the case of Marlin BB it is the Marlin contentID, in the case of CSPG-CI+ the value of this field is null). Otherwise, it may be null or undefined. • ParentalRatingCollection ratings – the parental rating value of the currently playing content. The ParentalRatingCollection object is defined in 7.9.6. • String DRMSystemID – optional argument that specifies the DRM System ID of the DRM system that generated the event as defined by element DRMSystemID in 4.4.3 of IEC 62766-3:2016. The value shall be null if the parental control is not enforced by a particular DRM system.			

7.14.6.2 Events

For the intrinsic events listed in the table below, a corresponding DOM event shall be generated in the following manner:

Intrinsic event	Corresponding DOM event	DOM Event properties
onParentalRatingChange	ParentalRatingChange	Bubbles: No Cancellable: No Context Info: contentID , ratings , DRMSystemID , blocked
onParentalRatingError	ParentalRatingError	Bubbles: No Cancellable: No Context Info: contentID , ratings , DRMSystemID .

The above DOM events are directly dispatched to the event target, and will not bubble nor capture. Applications should not rely on receiving these events during the bubbling or the capturing phase. The applications that use DOM event handlers shall call the `addEventListener()` method on the CEA-2014 A/V embedded object. The third parameter of `addEventListener`, i.e. "useCapture", will be ignored.

7.14.7 Extensions to A/V Control object for DRM rights errors

This subclause shall apply to OITF and/or server devices which have indicated support for DRM protection by providing one or more <drm> elements as specified in 9.3.11.

For notifying JavaScript about DRM licensing errors during playback of DRM protected A/V content through the CEA-2014 A/V Control object (as defined by as defined in 5.7.1 of CEA-2014-A), an OITF shall support the following intrinsic event property and corresponding DOM event, for the CEA-2014 A/V Control object.

<pre>function onDRMRightsError(Integer errorState, String contentID, String DRMSystemID, String rightsIssuerURL)</pre>
The function that is called: - whenever a rights error occurs for the A/V content (no licence, licence invalid), which has led to blocking consumption of the content; - whenever a rights change occurs for the A/V content (licence valid), which leads to unblocking the consumption of the content. This may occur during playback, recording or timeshifting of DRM protected AV content. The specified function is called with four arguments <code>errorState</code>, <code>contentID</code>, <code>DRMSystemID</code> and <code>rightsIssuerURL</code> which are defined as follows: <code>Integer errorState</code> – error code detailing the type of error: 0: no licence, consumption of the content is blocked; 1: invalid licence, consumption of the content is blocked; 2: valid licence, consumption of the content is unblocked. <code>String contentID</code> – the unique identifier of the protected content in the scope of the DRM system that raises the error (i.e. in the case of Marlin BB it is the Marlin contentID, in the case of CSPG-CI+ the value of this field is null). <code>String DRMSystemID</code> – DRMSystemID as defined by element DRMSystemID in Table 9 of IEC 62766-3:2016. For example, for Marlin, the DRMSystemID value is "urn:dvb:casystemid:19188". <code>String rightsIssuerURL</code> – optional element indicating the value of the rightsIssuerURL that can be used to non-silently obtain the rights for the content item currently being played for which this DRM error is generated, in cases whereby the rightsIssuerURL is known. Cases whereby the rightsIssuerURL is known include cases whereby the rightsIssuerURL has been extracted from the MPEG2 TS of the protected content, retrieved from the SD&S discovery record or from the associated BCG metadata. The corresponding rightsIssuerURL fields are defined in 4.2.2.4 of IEC 62766-7:2017 and in 4.4.3 of IEC 62766-3:2016 respectively. If different URLs are retrieved from the stream and the metadata, then the conflict resolution is implementation-dependent.

For the intrinsic events listed in the table below, a corresponding DOM event shall be generated in the following manner:

Intrinsic event	Corresponding DOM event	DOM Event properties
onDRMRightsError	DRMRightsError	Bubbles: No Cancellable: No Context Info: <code>errorState</code> , <code>contentID</code> , <code>DRMSystemID</code> , <code>rightsIssuerURL</code>

The above DOM event is directly dispatched to the event target, and will not bubble nor capture. Applications should not rely on receiving a `DRMRightsError` event during the bubbling or the capturing phase. Applications that use DOM event handlers shall call the

`addEventListener()` method on the CEA-2014 A/V Control object. The third parameter of `addEventListener`, i.e. "useCapture", will be ignored.

7.14.8 Extensions to A/V Control object for playing media objects

OITFs that support the API defined in 7.12, 7.4.7 and 7.4 shall support playback of downloaded and / or recorded content as follows:

- firstly, by setting the data attribute of the A/V Control object to that returned from the `uri` property of the Download or Recording object to be played back;
- using the method defined below on the A/V Control object, which shall be supported if any of the APIs defined in those subclauses are supported;
- where the HTML5 media elements are supported (see 9.3.18), by setting the `src` attribute of a `<video>` element to that returned from the `uri` property of the Download or Recording object to be played back.

Boolean <code>setSource(String id)</code>		
Description	Change the content item to be played by the A/V Control object to the content item represented by <code>id</code>. Valid IDs include: • download identifiers (i.e. corresponding to property <code>Download.id</code>); • recording identifiers (i.e. corresponding to property <code>Recording.id</code>); • CODAsset identifiers (i.e. corresponding to property <code>CODAsset.uid</code>). Support for each of these identifier types depends on the support for the individual subclauses in which they are defined. Depending on the type of content for <code>id</code>, the following semantics apply. If <code>id</code> is a download identifier, the OITF shall change the content item to be played to the downloaded item, or item being downloaded, for which the <code>Download.id</code> property (as defined in 7.4.5.2) corresponds to the given download identifier. The <code>type</code> attribute of the A/V Control object should change to the MIME type of the content item represented by the download identifier, or the MIME type of the content item corresponding to the first content item listed in the content access download descriptor in case the download identifier represents a download of a content access download descriptor that contains multiple <code><ContentItem></code> elements. The <code>data</code> attribute shall change to the same value as the download identifier. 7.14.2.6 defines more details about playback of downloaded content, and how it relates to the states of the A/V Control object. If <code>id</code> is a recording identifier, the OITF shall change the content item to be played to the recorded item, or item being recorded, for which the <code>Recording.id</code> property (as defined in 7.10.6.2) corresponds to the given recording identifier. The <code>type</code> attribute of the A/V Control object should change to the MIME type of the format in which the content was recorded. The <code>data</code> attribute shall change to the same value as the recording identifier. If <code>id</code> is a COD asset identifier, the OITF shall change the content item to be played to the CODAsset, for which the <code>CODAsset.uid</code> property (as defined in 7.5.7.2) corresponds to the given COD asset identifier. The <code>type</code> attribute of the A/V Control object should change to the MIME type of the COD Asset. The <code>data</code> attribute shall change to the same value as to COD asset identifier. If the content item represented by <code>id</code> can be accepted by the A/V Control object for playback, the method returns <code>true</code>. The method returns <code>false</code> if the item cannot be accepted for playback.	
Arguments	id	The ID of the item to be played.

7.14.9 Extensions to A/V Control object for UI feedback of buffering A/V content

7.14.9.1 General

If an OITF has indicated support for playback control as defined in 9.3.22 in its capability description, the A/V Control object defined in 5.7.1 of CEA-2014-A shall support the properties and methods defined in this subclause as follows:

- If the type attribute of the <playbackControl> element includes "buffering", then onReadyToPlay, readyToPlay, supportedStrategies, getAvailablePlayTime and setBufferingStrategy shall be supported.
- If the type attribute of the <playbackControl> element includes "has", then onRepresentationChange, onPeriodChange, availableRepresentationsBandwidth, currentRepresentation, maxRepresentation, minRepresentation and setRepresentationStrategy shall be supported.
- If the type attribute of the <playbackControl> element includes "dash", then onRepresentationChange, onPeriodChange, availableRepresentationsBandwidth, currentRepresentation, maxRepresentation, minRepresentation, setRepresentationStrategy, availableRepresentationIds and currentRepresentationId shall be supported.

7.14.9.2 Properties

Boolean readyToPlay
Property that can be used to inspect whether or not enough (as determined by the OITF) of the media after the current play position has been buffered to start playback. Returns true if enough data has been buffered. Returns false if not enough data has been buffered.
function onReadyToPlay()
The function that gets called when enough (as determined by the OITF) of the media after the current play position has been buffered to start/continue playback. This event shall be generated whenever there is a state transition between state 4 ("buffering") and state 1 ("playing"). The event shall also be generated at the moment that enough data has been buffered to start playback, whilst in state 2 ('paused').
function onRepresentationChange(Integer bandwidth, Integer position, String id)
When a HAS stream is being presented the function that shall be called when the stream changes Representation and the bandwidth is modified. The bandwidth relates to the bandwidth attribute in the Representation element of HTTP Adaptive Streaming manifest. The stream change relates to presentation changes and not necessarily how the stream is buffered. Note that multiple streams representing different qualities may be buffered and therefore is unreliable to indicate bandwidth change. The specified function shall be called with two arguments bandwidth and position which are defined as follows: • Integer bandwidth – the bandwidth associated with the new Representation. The unit used to represent bandwidth is the same as the bandwidth attribute in the manifest (i.e. bits per second (bps)). • Integer position – the position at which the transition to the new list of Representations associated with the new Period occurs. This should be in advance of the current play position but may not be. • String id – the identifier of the new Representation. This identifier shall be provided if the MPD includes an "id" attribute on the representation.

function onPeriodChange(IntegerCollection bandwidths, Integer position, StringCollection ids)

When a HAS stream or a DASH stream is being presented the function that shall be called immediately when a new manifest for a new Period is loaded, and the Representations and associated bandwidth are different to the current manifest. The bandwidth relates to the bandwidth attribute in the Representation element of the manifest.

Note that this should allow for an application to influence the buffering strategy before the position is reached but it not guaranteed. In which case, the buffering strategy may take effect after presentation of the new Period has been initiated. When the function is called it should be possible to modify the selected max and min Representation.

The specified function is called with two arguments bandwidths and position which are defined as follows:

- **IntegerCollection bandwidths** – the list of bandwidths associated with the new Period.
- **Integer position** – the position at which the transition to a new Representation occurs. This should be in advance of the current playPosition but may not be.
- **StringCollection ids** – These identifiers shall be provided if the MPD includes an "id" attribute on the representation.

readonly IntegerCollection availableRepresentationsBandwidth

When a HAS stream or a DASH stream is being presented, return an ordered list of the available Representations. Each Representation shall be identified by its respective bandwidth. Each Representation is identified by the bandwidth attribute in the Representation element of the MPD (as defined in IEC 62766-2-2 or ISO/IEC 23009-1).

readonly StringCollection availableRepresentationIds

If a stream is being presented that is described by an MPD and the MPD includes an **id** attribute on the representation then return an ordered list of the available Representations identified by the **id**. If the MPD does not include this attribute or if the player is a state other than play state 1 ('playing') then the value shall be undefined.

readonly Integer currentRepresentation

When a HAS stream or DASH stream is being presented return Representation that is being presented. The Representation is identified by the bandwidth attribute in the Representation element of the MPD (as defined in IEC 62766-2-2 or ISO/IEC 23009-1). The bandwidth is only available in play state 1 ('playing'), in other states the value is undefined.

readonly String currentRepresentationId

If a stream is being presented that is described by an MPD and the MPD includes an **id** attribute on the representation, then returns the id of the Representation that is being presented. If any other type of stream is being presented or if the player is in a state other than play state 1 ('playing') then the value is undefined.

readonly Integer maxRepresentation

Returns the maximum supported bandwidth from the **availableRepresentationsBandwidth** property. Note that calling the **setRepresentationStrategy()** method may modify the maximum bandwidth.

readonly Integer minRepresentation

Returns the minimum supported bandwidth from the **availableRepresentationsBandwidth** property. Note that calling the **setRepresentationStrategy()** method may modify the minimum bandwidth.

readonly StringCollection supportedStrategies	
The list of the supported buffering strategies. The supported strategy names are listed below. Note that other strategies may be supported.	
• "sustained_playback": if this strategy is supported, then the method setBufferingStrategy() shall be supported with this strategy. • "low_latency": if this strategy is supported, then the method setBufferingStrategy() shall be supported with this strategy. • "representation_strategy": if this strategy is supported, then the method setRepresentationStrategy() shall be supported.	

7.14.9.3 Methods

Integer getAvailablePlayTime(Boolean fromPlayPosition)		
Description	Returns how much content is available for playback. If argument fromPlayPosition has value true, this method returns an estimate of how much data in milliseconds is available in the buffer for play back after the current play position. If argument fromPlayPosition has value false, this method returns an estimate of the total buffer length in milliseconds (i.e. this includes all data available in the buffer before and after the current play position).	
Arguments	fromPlayPosition	Indicates whether the available play time should be calculated from the current play position onwards, or from the start of the buffer.

Boolean setBufferingStrategy(String name)		
Description	Request to change the buffering strategy to that given by name. This method can be called during any play state, including play state 1 ('playing'). This method returns true if the buffering strategy has been successfully changed to the preferred buffering strategy. The method returns false if the buffering strategy has not been successfully changed. If the OITF does not distinguish between the two modes, the method returns false.	
Arguments	name	The name of the requested buffering strategy. Valid values include: "sustained_playback": this is the default strategy, whereby the incoming video stream should be rendered with as little hickups or lost frames as possible. This means that the buffering threshold for triggering an onReadyToPlay event is chosen to be sufficiently large to deal with variations in network throughput. "low_latency": this is a strategy whereby the incoming video stream should be rendered with an as-low-as-possible latency between receiving the content and the actual playback of the content. This means that buffering threshold for triggering an onReadyToPlay event needs to be made sufficiently small in order to playback the content as soon as possible after it has been received. These values are not case sensitive. The default strategy if the method is not called is "sustained_playback".

Boolean setRepresentationStrategy (Integer maxBandwidth, Integer minBandwidth, Integer position)		
Description	Request to change the strategy for selecting which Representation to use from the specified position for HTTP Adaptive Streaming or MPEG DASH. The indicated bandwidth represents the maximum and minimum bandwidth to be allowed. Representations outside of the upper and lower limits shall not be selected. If data has already been fetched outside these limits, then there is no requirement to discard that data. This method can be called during any play state, including play state 1 ('playing'). Only one change in strategy is in effect and any previous strategy that has not taken effect is overwritten. This method returns true if the strategy has been successfully changed. The method returns false if the buffering strategy has not been successfully changed. The value of maxBandwidth shall be greater than minBandwidth otherwise the method shall return false. The range between the bandwidth from maxBandwidth and minBandwidth shall allow for at least one of the values from property availableRepresentations otherwise the method shall return false. If the Period changes and no Representations remain that are within the set Representation strategy then the maxBandwidth and minBandwidth shall be reset to undefined. In order to avoid this from occurring the max and/or min bandwidth have to be immediately modified when the onPeriodChange function is called.	
Arguments	maxBandwidth	The maximum bandwidth allowed for the presentation of adaptive content. If value set to undefined the limit is set by the OITF.
	minBandwidth	The minimum bandwidth allowed for the presentation of adaptive content. If value set to undefined the limit is set by the OITF.
	position	This argument is optional. If present it indicates the position at which the new Representation strategy shall be applied. The position should be the same as the position returned in onPeriodChange() to make for a smooth transition to a new strategy.

7.14.9.4 Events

For the intrinsic events listed in the table below, a corresponding DOM event shall be generated in the following manner:

Intrinsic event	Corresponding DOM event	DOM Event properties
onReadyToPlay	ReadyToPlay	Bubbles: No Cancellable: No Context Info: None
onRepresentationChange	RepresentationChange	Bubbles: No Cancellable: No Context Info: bandwidth, position
onPeriodChange	PeriodChange	Bubbles: No Cancellable: No Context Info: bandwidths, position

These DOM events are directly dispatched to the event target, and will not bubble nor capture. Applications should not rely on receiving these events during the bubbling or the capturing phase. Applications that use DOM event handlers shall call the **addEventListener()** method on the CEA-2014 A/V Control object. The third parameter of **addEventListener**, i.e. "useCapture", will be ignored.

7.14.10 DOM events for A/V Control object

To make the A/V Control object as defined in CEA-2014-A in line with the other scripting objects in Clause 7, for the intrinsic events listed in the table below, a corresponding DOM event shall be generated in the following manner:

Intrinsic event	Corresponding DOM event	DOM Event properties
onfocus	focus (as defined in 5.2.1.2 of the DOM Level 3 Events specification as referenced in IEC 62766-5-2)	Bubbles: No Cancellable: No Context Info: None
onblur	blur (as defined in 5.2.1.2 of the DOM Level 3 Events specification as referenced in IEC 62766-5-2)	Bubbles: No Cancellable: No Context Info: None
onPlayStateChange	PlayStateChange	Bubbles: No Cancellable: No Context Info: None
onFullScreenChange	FullScreenChange	Bubbles: No Cancellable: No Context Info: None

These DOM events are directly dispatched to the event target, and will not bubble nor capture. Applications should not rely on receiving these events during the bubbling or the capturing phase. Applications that use DOM event handlers shall call the `addEventListener()` method on the CEA-2014 A/V Control object. The third parameter of `addEventListener`, i.e. "useCapture", will be ignored.

When handling `PlayStateChange` events, since the property "playState" of the A/V Control object always returns the current play state, there are a number of considerations:

- when accessing the `playState` property inside a `PlayStateChange` event handler, its value will be the current state of the related media object that may be different from the state transition that caused the handler to be called;
- the `playState` property may change value during the execution of the `PlayStateChange` event handler;
- for an A/V Control object there is no way to detect which state transition caused the event handler to be executed.

7.14.11 Playback of memory audio

7.14.11.1 General

Subclause 7.14.11 describes how an A/V Control object can be used for the playback of audio from memory.

7.14.11.2 Usage of CE-HTML tags

The A/V Control object shall be used to play audio clips from memory. The value of the A/V Control object's `type` attribute shall be set to one of the values defined in 9.3.1 of IEC 62766-2-1:2016. The `<object>` element representing the A/V Control object may contain `<param>` elements to set the value of parameters affecting the playback of the clip. For audio from memory, valid parameters are:

- `cache` – a value of "true" indicates that the audio clip should be played from memory. This parameter shall be included for all clips to be played from memory. For formats which can not be played from memory, or for values of the parameter other than "true", this parameter shall have no effect. The default value of this parameter shall be "false".
- `loop` – indicates the number of times the audio clip shall be played when `play()` is called. The value shall be positive integers or the string "infinite", which shall play the audio clip continuously until `stop()` is called or the data property is set to null. The default value of this parameter shall be "1".

Simultaneous playback of multiple audio clips from memory, or simultaneous playback of audio clips from memory and streaming audio or video presentation shall follow the behaviour described in 4.5.6.

7.14.11.3 Usage of the DOM interface

For A/V Control objects used to play audio from memory, the following properties and methods shall be supported:

- the properties `data`, `playState`, `error` and `onPlayStateChange`, as defined in Req. 5.7.1.f of CEA-2014-A;
- the methods `play()` and `stop()`, as defined in Req. 5.7.1.f of CEA-2014-A.

When the `play()` method is called, if a `<param>` element as described above is present where the `cache` parameter is set to the value "true", the OITF shall:

- attempt to pre-load the audio clip specified by the value of its `data` property and play the audio clip from memory; if the terminal cannot pre-load the audio clip due to insufficient memory, the terminal shall play the clip as streaming audio;
- attempt to retain the audio clip in its cache once playback has finished, until the A/V Control object's `data` property is modified or the A/V Control object is destroyed.

If the A/V Control object's `data` property refers to a file in a format other than those listed in 9.3.1 of IEC 62766-2-1:2016, the A/V Control object shall not attempt to play the file from memory.

The `<param>` element as defined in 7.14.11.2 shall be made accessible through a DOM `HTMLParamElement` object.

7.14.11.4 Example usage

The following HTML document shows an example of a script to start the playback of memory audio:

```
<head>
:
<script type="text/javascript">
    function startBGM() {
        document.getElementById("aid1").play(1);
    }
:
</script>
</head>
<body>
<object                                type="audio/mp4"                                id="aid1"
data="http://www.avsource.com/audio/bgm.aac">
    <param name="cache" value="true"/>
    <param name="loop" value="infinite"/>
</object>
:
<div id="btn1" onclick="startBGM()"></div>
:
</body>
```

The following HTML document shows an example of a script to stop the playback of memory audio:

```
<head>
:
<script type="text/javascript">
    function stopBGM() {
        document.getElementById("aid1").stop();
    }
:
</script>
```

```

</head>
<body>
<object                                type="audio/mp4"                                id="aid1"
data="http://www.avsource.com/audio/bgm.aac">
<param name="cache" value="true"/>
<param name="loop" value="infinite"/>
</object>
:
<div id="btn2" onclick="stopBGM()"></div>
:
</body>

```

7.14.12 Extensions to A/V Control object for media queuing

7.14.12.1 General

The following additional method shall be supported on the audio object and video object defined in 5.7.1 of CEA-2014-A.

Boolean queue (String uri)			
Description	Queue the media referred to by uri for playback after the current media item has finished playing. If a media item is already queued, uri will not be queued for playback and this method will return false. If the item is queued successfully, this method returns true. If no media is currently playing, the queued item will be played immediately. If uri is null, any currently queued item will be removed from the queue and this method will return true. If an A/V Control object is an audio object as defined by 5.7.1.b.1 of CEA-2014-A, then queued media items shall only contain audio. If an A/V Control object is a video object as defined by 5.7.1.b.2 of CEA-2014-A, then queued media items shall always contain video and may also contain audio and other media components. Applications should ensure the value of uri refers to a media format appropriate to the instance of the A/V Control object. There are no requirements on queued media items apart from the preceding ones. Specifically, there is no requirement for the MIME type of queued media items to match the contents of the type attribute of the object element. If the value of the type attribute differs from the MIME type of the queued media item, the MIME type shall take precedence. When the current media item has finished playing, the A/V Control object shall transition to the finished state, update the value of the data property with the URL of the queued media item and automatically start playback of the queued media item. The A/V Control object may transition to the connecting or buffering states (and generate the necessary PlayStateChange events) before entering the playing state when the queued media item is being presented. Implementations may pre-buffer data from the queued URL before the current media item has finished playing in order to reduce the delay between items. If the queued media item can be played without transitioning to the connecting or buffering states, then the A/V Control object shall generate a PlayStateChange event to the playing state to indicate that the queued media item has started playing. If playback of the current media item is stopped using the stop() method, or if the data and/or type property is modified, the queued media item shall not be played and the queued media item shall be discarded as if no item was queued. Play speed is not affected by transitioning between the current and queued media item. To avoid race conditions when queuing multiple items for playback, applications should wait for the currently queued item to begin playback before queuing subsequent items, e.g. by queuing the subsequent item when the A/V Control object transitions to the connecting, buffering or playing state for the currently queued item.		
Arguments	<table> <tr> <td>uri</td><td>The media item to be queued, or null to remove the currently-queued item.</td></tr> </table>	uri	The media item to be queued, or null to remove the currently-queued item.
uri	The media item to be queued, or null to remove the currently-queued item.		

7.14.12.2 URI support and the queue method

All URIs that are supported by the OITF as the data property of the object element shall be supported as the uri argument to the queue method. Furthermore:

- URIs that directly reference a media item shall be supported both with and without a media fragment (as defined in 8.3.2);
- using a URI referencing a content access streaming descriptor as the uri argument shall be supported. In such a content access streaming descriptor, all URIs for non-local

content (i.e. excluding local PVR and Download) that are supported as the `uri` argument to the `queue()` method shall be supported in the `<contentURL>` element.

For example, an OITF supporting HTTP streaming and downloaded content but none of local PVR, HTTP adaptive streaming or RTSP streaming will support the following as the `uri` element:

- HTTP URL directly referencing the complete media item (i.e. without a fragment);
- HTTP URL directly referencing part of the media item (i.e. with a media time fragment);
- HTTP URL referencing a content access streaming descriptor where the `<contentURL>` element is an HTTP URL referencing the complete media item (i.e. without a fragment);
- HTTP URL referencing a content access streaming descriptor where the `<contentURL>` element is an HTTP URL referencing part of the media item (i.e. with a media time fragment);
- private URI directly identifying a complete downloaded content item (i.e. without a fragment);
- private URI directly referencing part of a downloaded content item (i.e. with a fragment).

Additionally, the OITF shall support all combinations of all URIs that can be used as the `uri` argument with all URIs that can be used as the data property of the object element.

7.14.12.3 Implementation Requirements on the Queue Method

When the queue method is used, once the last required data of the current content item has been read, the OITF shall start reading from the queued content item without waiting for the last required data of the current content item to be decoded and presented. If the current content item was identified by a URL without a fragment or with a fragment not including an end time, then the last required data is the last data of the content item. If the current content item was identified by a URL with a fragment including an end time, then the last required data is the last data needed to decode and present the frame corresponding to that end time.

7.14.13 Extensions to A/V Control object for volume control

7.14.13.1 Methods

The following additional method shall be supported on the audio object and video object defined in 5.7.1 of CEA-2014-A.

Integer <code>getVolume()</code>	
Description	Returns the actual volume level set; for systems that do not support individual volume control of players, this method will have no effect and will always return 100.

7.14.14 Extensions to A/V Control object for resource management

7.14.14.1 General

Subclause 7.14.14 defines APIs related to resource capabilities allocated to the A/V Control object.

7.14.14.2 Constants

Name	Value	Use
STATIC_ALLOCATION	1	Scarce resources are allocated at instantiation time
DYNAMIC_ALLOCATION	2	Scarce resources are allocated to the object as required

7.14.14.3 Properties

readonly StringCollection <code>playerCapabilities</code>
The list of media formats that are supported by the object. Each item shall contain a format label according to IEC 62766-2-1.
If scarce resources are not claimed by the object, the value of this property shall be <code>null</code> .

readonly Integer <code>allocationMethod</code>
Returns the resource allocation method currently in use by the object. Valid values as defined in 7.14.14.2 are:
• <code>STATIC_ALLOCATION</code> • <code>DYNAMIC_ALLOCATION</code>

7.15 Miscellaneous APIs

7.15.1 The application/oipfMDTF embedded object

7.15.1.1 General

If an OITF has indicated support for the multicast delivery terminating function (MDTF) (i.e., `<mdtf>true</mdtf>`) as defined in 9.3.16 in its capability description, the OITF shall support MDTF through the use of the following non-visual object:

```
<object type="application/oipfMDTF" />
```

The MDTF API provides the necessary JavaScript methods to indicate to the MDTF what FLUTE multicast channel it should join, and what tags it should listen for on those channels.

7.15.1.2 Properties

function **onFLUTEListenerResult**(String multicastAddress, Integer resultMsg)

This function is called with return result from the methods **addFLUTEListener** and **removeFLUTEListener**.

The specified script function is called with two arguments – **multicastAddress** and **resultMsg**.

String multicastAddress – The multicast address associated with the callback.

Integer resultMsg – result message. Valid values include:

Result message	Description	Semantics
0	Successful	The action performed by the underlying functionality was successful.
1	Unknown error	The action performed by the underlying functionality failed because an unspecified error occurred.
2	Invalid multicast address	The multicast address is not valid, e.g. bad syntax or out of range.
3	Multicast address does not exist	The multicast address does not exist in the listener table.
4	No resources	There was not enough resources in the OITF to join the multicast address (only valid for addFLUTEListener()).

7.15.1.3 Methods

void addFLUTEListener(String multicastAddress)		
Description	This method adds a FLUTE channel listener in the OITF. The result from this method is sent to the callback method onFLUTEListenerResult .	
Arguments	multicastAddress	The multicast address that the OITF should join in order to listen.

void addFLUTEListenerTags(String multicastAddress, String tags, function downloadCallBack)		
Description	This method adds tags that the FLUTE listener should listen for. The result from this method is sent to the callback method onFLUTEListenerResult .	
Arguments	multicastAddress	The multicast address that the OITF should join in order to listen.
	tags	A comma-separated list of tags that the OITF should listen for on the FLUTE channel.
	downloadCallBack	Optional. This callback function is called when an object has been downloaded. The arguments to this function are the Content Location URI of the downloaded object and the Content-Type.

StringCollection getFLUTEListeners()	
Description	Returns a collection of multicast addresses for the FLUTE channels that the OITF listens to.

String getTags(String multicastAddress)	
Description	Returns a comma-separated list of the tags associated with a particular multicast address.

void removeFLUTEListener(String multicastAddress)		
Description	Removes the associated listener. The result from this method is sent to the callback method onFLUTEListenerResult .	
Arguments	multicastAddress	The multicast address that the OITF should leave.

7.15.1.4 Events

For the intrinsic events listed in the table below, a corresponding DOM event shall be generated in the following manner:

Intrinsic event	Corresponding DOM event	DOM Event properties
onFLUTEListenerResult	FLUTEListenerResult	Bubbles: No Cancellable: No Context Info: multicastAddress, resultMsg

The above DOM event is directly dispatched to the event target, and will not bubble nor capture. Applications should not rely on receiving a FLUTEListenerResult event during the bubbling or the capturing phase. Applications that use DOM event handlers shall call the **addEventListener()** method on the application/oipfMDTF object. The third parameter of **addEventListener**, i.e. "useCapture", will be ignored.

7.15.2 The application/oipfStatusView embedded object

7.15.2.1 Overview of download status

7.15.2.1.1 General

The following embedded objects allow a visualization of the native download manager to be included as part of the UI coming from a (third party) server, without the need for any security model, and without compromising security and privacy.

An OITF shall support the application/oipfStatusView embedded object. This is a visual object that can be included in a HTML document, and is subject to the following CSS properties: width, height, position, float, top, left, right, bottom, vertical-align, padding, and padding-* properties, margin, and margin-* properties, border, and border-* properties, visibility, and display. This embedded object shall provide an overall consistent graphical view of the status of the current downloads, the content that has been downloaded, and/or the content that has been recorded, as denoted by the states:

- "list_of_recent_downloads": shows the progress of the most recently started downloads, with the amount of items shown as specified by a <param> element with the name "nritems";
- "list_of_downloaded_content": shows the list of items that have been successfully downloaded, with the amount of items shown as specified by a <param> element with the name "nritems".

The object shall support a <param> element with the name "state", which indicates the state that shall be visualized inside the object. An OITF that has indicated support for downloading content in its capability description (i.e. <download>true</download>) shall at least support the monitor states "list_of_recent_downloads" and "list_of_downloaded_content". An OITF may support the visualization of additional states. An OITF shall silently ignore a request to visualize a state that it does not support; if this results in no state information being visualized at all (because the each <param> element with name state referred to a non-supported state), the application/oipfStatusView object shall not be visualized and the object will have CSS width and height values of 0.

The object shall support a <param> element with the name "nritems", which indicates the number of items that should be shown for the given state.

The object shall also support the inclusion of style hints through <param> elements. At least the "background-color" and "font-size" style hints shall be supported using the syntax defined by CSS 2.1. An OITF may support additional style hints in addition to "background-color" and "font-size". Additional style hints shall also follow the CSS 2.1 syntax. An OITF shall silently ignore any style hints that it does not support.

Next to these parameters, the object shall support methods "getMinimumItemwidth()" and "getMinimumItemHeight()" as defined in 7.15.2.1.2.

Example usage:

```
<object id="d1" type="application/oipfStatusview" width="200" height="100">
  <param name="state" value="list_of_recent_downloads"/>
  <param name="nritems" value="2"/>
  <param name="background-color" value="black"/>
  <param name="font-size" value="16px"/>
</object>
```

NOTE This object is intended to allow services to link in to the privileged functionality of accessing privacy sensitive download information, without the need for certificates and privileged access requests. In certain managed deployments this may not be sufficient. The application/oipfDownloadManager API described in 7.4.4 provides more extensive APIs which provide JavaScript control for a service platform provider over such highly privileged functionality.

7.15.2.1.2 Methods

Integer getMinimumItemWidth (String state)		
Description	Returns the minimum width needed for rendering the name, status and other data of the downloaded items for the given state (e.g. "list_of_recent_downloads").	
Arguments	state	The state for which the visualization is requested. This is one of the strings that are defined for <param> element with the name "state" (e.g. "list_of_recent_downloads").

Integer getMinimumItemHeight (String state)		
Description	Returns the minimum height needed for rendering the name, status and other data of the downloaded items for the given state (e.g. "list_of_recent_downloads").	
Arguments	state	The state for which the visualization is requested. This is one of the strings that are defined for <param> element with the name "state" (e.g. "list_of_recent_downloads").

7.15.2.2 Overview of recordings

An OITF that has indicated support for control of its recording functionality by a server (i.e., <record>true</record>) shall support the application/oipfStatusView embedded object defined in 7.15.2.1, for which it shall at least support the following additional monitor state:

- "list_of_recorded_content": shows the list of items that have been recorded or that are currently being recorded, with the amount of items shown as specified by <param> element with the name "nrItems".

NOTE This object is intended to allow services to link in to highly privileged functionality, without the need for certificates and privileged access requests. In certain managed deployments this may not be sufficient. Therefore, 7.10.5 defines more extensive APIs which provide JavaScript control for a service platform provider over such highly privileged functionality.

7.15.3 The application/oipfCapabilities embedded object

7.15.3.1 General

The OITF shall support following non-visual embedded object with the mime type "application/oipfCapabilities".

7.15.3.2 Properties

readonly Document xmlCapabilities
Returns the OITF's capability description as an XML Document object using the syntax as defined in Annex F without using any namespace definitions.

readonly Number extraSDVideoDecodes
This property holds the number of possible additional decodes for SD video. Depending on the current usage of system resources this value may vary. The value of this property is likely to change if an HD video is started.
Adding an A/V Control object or video/broadcast object may still fail, even if extraSDVideoDecodes is larger than 0. For A/V Control objects, in case of failure the play state for the A/V Control object shall be set to 6 ('error') with a detailed error code of 3 ('insufficient resources'). For video/broadcast objects, in case of failure, the play state of the video/broadcast object shall be set to 0 ('unrealized') with a detailed error code of 11 ('insufficient resources').

readonly Number <code>extraHDVideoDecodes</code>
This property holds the number of possible additional decodes for HD video. Depending on the current usage of system resources this value may vary. The value of this property is likely to change if an SD video is started. Adding an A/V Control object or <code>video/broadcast</code> object may still fail, even if <code>extraHDVideoDecodes</code> is larger than 0. For A/V Control objects, in case of failure the play state for the A/V Control object shall be set to 6 ('error') with a detailed error code of 3 ('insufficient resources'). For <code>video/broadcast</code> objects, in case of failure the play state of the <code>video/broadcast</code> object shall be set to 0 ('unrealized') with a detailed error code of 11 ('insufficient resources').

7.15.3.3 Methods

Boolean <code>hasCapability(String profileName)</code>		
Description	Check if the OITF supports the passed capability. Returns <code>true</code> if the OITF supports the passed capability, <code>false</code> otherwise.	
Arguments	<code>profileName</code>	An OIPF base UI profile string or a UI Profile name fragment string as defined in 9.2. Examples of valid <code>profileName</code>: "OITF_HD_UIPROF" or "+PVR".

7.15.4 The Navigator class

The navigator object is defined by the HTML5 specification as referenced by IEC 62766-5-2.

7.15.5 Debug print API

The following method is available on the script's global object as defined in the HTML5 specification as referenced by IEC 62766-5-2.

void <code>debug(DOMString arg)</code>		
Description	Let the application developer print debug information on the debug output (for example, a console, a serial link or a file). The means to access this debug output is outside the scope of this document and implementation-dependent. A line feed character shall not be inserted automatically at the end of the string by the implementation. EXAMPLE: <pre>debug("[APP] value = " + value + "\n");</pre>	
Arguments	<code>arg</code>	String to print on the debug output.

7.16 Shared Utility classes and features

7.16.1 Base collections

7.16.1.1 The StringCollection class

```
typedef Collection<String> StringCollection
```

The `StringCollection` class represents a collection of `String` objects. See Annex I for the definition of the collection template.

7.16.1.2 The IntegerCollection class

```
typedef Collection<Integer> IntegerCollection
```

The `IntegerCollection` class represents a collection of `Integer` values. See Annex I for the definition of the collection template.

7.16.2 The Programme class

7.16.2.1 General

The Programme class represents an entry in a programme schedule.

NOTE As described in the `record ( Programme programme )` method of the `application/oipfRecordingScheduler` object, only the `programmeID` property of the programme object is used to determine the programme or series that will be recorded. The other properties are solely used for annotations of the (scheduled) recording with programme metadata. The use of these metadata properties is optional. If such programme metadata is provided, it is retained in the `ScheduledRecording` object that is returned if the recording of the programme was scheduled successfully.

7.16.2.2 Constants

Name	Value	Use
ID_TVA_CRID	0	Used in the <code>programmeIDType</code> property to indicate that the programme is identified by its TV-Anytime CRID (Content Reference Identifier).
ID_DVB_EVENT	1	Used in the <code>programmeIDType</code> property to indicate that the programme is identified by a DVB URL referencing a DVB-SI event as enabled by 5.2.4 of IEC 62766-3:2016. Optional.
ID_TVA_GROUP_CRID	2	Used in the <code>programmeIDType</code> property to indicate that the Programme object represents a group of programmes identified by a TV-Anytime group CRID.

7.16.2.3 Properties

String name
The short name of the programme, e.g. 'Star Trek: DS9'.

String longName
The long name of the programme, e.g. 'Star Trek: Deep Space Nine'. If the long name is not available, this property will be undefined .

String description
The description of the programme, e.g. an episode synopsis. If no description is available, this property will be undefined .

String longDescription
The long description of the programme. If no description is available, this property will be undefined .

Integer startTime
The start time of the programme, measured in seconds since midnight (GMT) on 1/1/1970.

Integer duration
The duration of the programme (in seconds).

String channelID
The identifier of the channel from which the broadcasted content is to be recorded. Specifies either a <code>ccid</code> or <code>ipBroadcastID</code> (as defined by the <code>Channel</code> object in 7.13.12).

Integer episode
The episode number for the programme if it is part of a series. This property is undefined when the programme is not part of a series or the information is not available.

Integer totalEpisodes
If the programme is part of a series, the total number of episodes in the series. This property is undefined when the programme is not part of a series or the information is not available.

readonly Boolean is3D
Flag indicating whether the programme has 3D video.

String programmeID
The unique identifier of the programme or series, e.g., a TV-Anytime CRID (Content Reference Identifier).

Integer programmeIDType
The type of identification used to reference the programme, as indicated by one of the ID_* constants defined above.

readonly String IMI
The TV-Anytime Instance Metadata ID for this programme.

readonly ParentalRatingCollection parentalRatings
A collection of parental rating values for the programme for zero or more parental rating schemes supported by the OITF. For instances of the Programme class created by the createProgramme() method defined in 7.10.2.2, the initial value of this property (upon creation of the Programme object) is an instance of the ParentalRatingCollection object (as defined in 7.9.6) with length 0. Parental rating values can be added to this empty readonly parental rating collection by using the addParentalRating() method of the ParentalRatingCollection object. The ParentalRatingCollection is defined in 7.9.6. The related ParentalRating and ParentalRatingScheme objects are defined in 7.9.5 and 7.9.3 respectively. For instances of the Programme class returned through the metadata APIs defined in 7.12 or through the programmes property of the video/broadcast object defined in 7.13.4, the initial value of this property shall include the parental rating value(s) carried in the metadata or DVB-SI entry describing the programme, if this information is included. Note that if the service provider specifies a certain parental rating (e.g. PG-13) through this property and the actual parental rating extracted from the stream says that the content is rated PG-16, then the conflict resolution is implementation dependent.

readonly StringCollection groupCRIDs
The group CRIDs associated with this programme.

7.16.2.4 Metadata extensions to Programme

7.16.2.4.1 General

The OITF shall extend the **Programme** class defined in 7.16.2 with the properties and methods described in 7.16.2.4.2 and 7.16.2.4.3.

This subsubclause shall apply for OITFs that have indicated <clientMetadata> with value "true" and a "type" attribute with values "bcg", "eit-pf" or "dvb-si" as defined in 9.3.8 in their capability profile.

7.16.2.4.2 Properties

readonly channel channel
Reference to the broadcast channel where the programme is available.
The value of this field is derived from the serviceIDref attribute of the Schedule element that refers to this programme.

readonly Boolean blocked															
Flag indicating whether the programme is blocked due to parental control settings or conditional access restrictions.															
The blocked and locked properties work together to provide a tri-state flag describing the status of a programme. This can best be described by the following table:															
<table><tr><th>Description</th><th>blocked</th><th>locked</th></tr><tr><td>No parental control applies.</td><td>false</td><td>false</td></tr><tr><td>Item is above the parental rating threshold (or manually blocked); no PIN has been entered to view it and so the item cannot currently be viewed.</td><td>true</td><td>true</td></tr><tr><td>Item is above the parental rating threshold (or manually blocked); the PIN has been entered and so the item can be viewed.</td><td>true</td><td>false</td></tr><tr><td>Invalid combination – OITFs shall not support this combination</td><td>false</td><td>true</td></tr></table>	Description	blocked	locked	No parental control applies.	false	false	Item is above the parental rating threshold (or manually blocked); no PIN has been entered to view it and so the item cannot currently be viewed.	true	true	Item is above the parental rating threshold (or manually blocked); the PIN has been entered and so the item can be viewed.	true	false	Invalid combination – OITFs shall not support this combination	false	true
Description	blocked	locked													
No parental control applies.	false	false													
Item is above the parental rating threshold (or manually blocked); no PIN has been entered to view it and so the item cannot currently be viewed.	true	true													
Item is above the parental rating threshold (or manually blocked); the PIN has been entered and so the item can be viewed.	true	false													
Invalid combination – OITFs shall not support this combination	false	true													

Integer showType								
Flag indicating the type of show (live, first run, rerun, etc.).								
The value of this property is determined by the child elements of the programme's BroadcastEvent or ScheduleEvent element from the Program Location Table. Values are determined as follows:								
<table><tr><th>Value</th><th>Description</th></tr><tr><td>1</td><td>The programme is live; indicated by the presence of a Live element with a value attribute set to true.</td></tr><tr><td>2</td><td>The programme is a first-run show; indicated by the presence of a FirstShowing element with a value attribute set to true.</td></tr><tr><td>3</td><td>The programme is a rerun; indicated by the presence of a Repeat element with a value attribute set to true.</td></tr></table>	Value	Description	1	The programme is live; indicated by the presence of a Live element with a value attribute set to true.	2	The programme is a first-run show; indicated by the presence of a FirstShowing element with a value attribute set to true.	3	The programme is a rerun; indicated by the presence of a Repeat element with a value attribute set to true.
Value	Description							
1	The programme is live; indicated by the presence of a Live element with a value attribute set to true.							
2	The programme is a first-run show; indicated by the presence of a FirstShowing element with a value attribute set to true.							
3	The programme is a rerun; indicated by the presence of a Repeat element with a value attribute set to true.							
If none of the above conditions are met, the default value of this field shall be 2.								

Boolean subtitles
Flag indicating whether subtitles or closed-caption information is available.
This flag shall be true if one or more BCG CaptionLanguage elements are present in this programme's description, false otherwise.

Boolean isHD
Flag indicating whether the programme has high-definition video.
This flag shall be true if a VerticalSize element is present in the programme's description and has a value greater than 576, false otherwise.

Integer audioType	
Bitfield indicating the type of audio that is available for the programme.	
The value of this field is determined by the NumOfChannels elements in a programme's A/V attributes. Values are determined as follows:	
Value	Description
1	A mono audio stream is available (at least one AvAttributes.AudioAttributes element is present which has a child NumOfChannels element whose value is 1).
2	A stereo audio stream is available (at least one AvAttributes.AudioAttributes element is present which has a child NumOfChannels element whose value is 2).
4	A multi-channel audio stream is available (at least one AvAttributes.AudioAttributes element is present which has a child NumOfChannels element whose value is greater than 2).
For programmes with multiple audio streams, these values may be ORed together.	

Boolean isMultilingual
Flag indicating whether more than one audio language is available for the programme.
This flag shall be true if more than one BCG Language element is present in the programme's description, false otherwise.

StringCollection genre
A collection of genres that describe this programme.
The value of this field is the concatenation of the values of any Name elements that are children of Genre elements in the programme's description.

readonly Boolean hasRecording
Flag indicating whether the Programme has a recording associated with it (either scheduled, in progress, or completed).

StringCollection audioLanguages
Supported audio languages, indicated by their ISO 639-2 language codes as defined in ISO 639-2.

StringCollection subtitleLanguages
Supported subtitle languages, indicated by their ISO 639-2 language codes as defined in ISO 639-2.

readonly Boolean locked
Flag indicating whether the current state of the parental control system prevents the programme from being viewed (e.g. a correct parental control PIN has not been entered to allow the programme to be viewed).

7.16.2.4.3 Methods

String getField(String fieldId)		
Description	Get the value of the field referred to by fieldId that is contained in the metadata for this programme. If the field does not exist, this method shall return undefined .	
Arguments	fieldId	The name of the field whose value shall be retrieved.

7.16.2.5 DVB-SI extensions to Programme

The following method shall be added to the Programme object, if the OITF has indicated support for accessing DVB-SI information, by giving the value "true" to element <clientMetadata> and the value "dvb-si" or "eit-pf" to the "type" attribute of that element as defined in 9.3.8 in their capability profile.

StringCollection getSIDescriptors (Integer descriptorTag, Integer descriptorTagExtension, Integer privateDataSpecifier)		
Description	Get the contents of the descriptor specified by descriptorTag from the DVB SI EIT programme's descriptor loop. If more than one descriptor with the specified tag is available for the given programme, the contents of all matching descriptors shall be returned in the order the descriptors are found in the stream. The descriptor content bytes shall be encoded in a string whose characters shall be restricted to the ISO Latin-1 character set. Each character in the string represents a byte of a DVB-SI descriptor, such that a byte at position "i" in the descriptor is equal the Latin-1 character code of the character at position "i" in the string. Described in the syntax of JavaScript: let desc[] be the byte array of a descriptor, in which desc[0] is the descriptor_tag, then, the returned string (retval in the example below) is its equivalent string, if: <pre>desc.length==retval.length and for each integer i: 0<=i<desc.length holds desc[i] == retval.charCodeAt(i).</pre> If the descriptor specified by descriptorTag and (optionally) descriptorTagExtension and privateDataSpecifier does not exist, or if the metadata for this programme was retrieved from a source other than DVB-SI, this method shall return null. If metadata for this programme has not yet been retrieved, this method shall return undefined. If the OITF supports the application/oipfSearchManager object as defined in subclause 7.12.2, the OITF shall notify applications of the availability of additional metadata via MetadataSearch events targeted at the application/oipfSearchManager object used to retrieve the programme metadata.	
Arguments	descriptorTag	The descriptor tag as specified by ETSI EN 300 468
	descriptorTagExtension	An optional argument giving the descriptor tag extension as specified by ETSI EN 300 468 This argument is mandatory when descriptorTag is 0x7f and ignored in all other cases.
	privateDataSpecifier	An optional argument giving the private_data_specifier as specified by ETSI EN 300 468. If this argument is present, only descriptors related to the identified specifier will be returned.

7.16.2.6 Recording extensions to Programme

The OITF shall support the following extensions to the Programme class.

Clients supporting the recording management APIs defined in this subclause shall indicate this by adding the attribute "manageRecordings" to the <recording> element with a value unequal to 'none' in the client capability description as defined in 9.3.4.

The functionality as described in this subclause is subject to the security model of Clause 10.

readonly ScheduledRecording recording
If available, this property represents the recording associated with this programme (either scheduled, in-progress or completed). Has value undefined if this programme has no scheduled recording associated with it.

7.16.3 The ProgrammeCollection class

```
typedef          collection<Programme>          ProgrammeCollection
```

The `ProgrammeCollection` class represents a collection of `Programme` objects. See Annex I for the definition of the collection template.

7.16.4 The DiscInfo class

7.16.4.1 General

The `DiscInfo` class provides details of the storage usage and capacity in the OITF.

A `DiscInfo` instance obtained from the `oipfDownloadManager` provides reports relating to downloads. A `DiscInfo` instance obtained from `oipfRecordingScheduler` provides reports relating to recordings. If recordings and downloads use the same pool of storage space (e.g. disc partition), `DiscInfo` instances obtained via either route would have the same values for the properties. If recordings and downloads use different pools of storage space (e.g. different disc partitions) the `DiscInfo` instances obtained by each route would report the correct values for the route in which they were obtained.

7.16.4.2 Properties

readonly Integer free
The space (in megabytes) available on the storage device.
readonly Integer total
The total capacity (in megabytes) of the storage device. Depending upon the system, free may be less than total as some of the disc space may be used for management purposes.
readonly Integer reserved
The space (in megabytes) reserved.

7.16.5 Extensions for playback of selected media components

7.16.5.1 General

This subclause defines APIs for the selection of specific A/V components for playback.

NOTE The term component can correspond to MPEG_2 components, but is not restricted to that.

7.16.5.2 Media playback extensions

7.16.5.2.1 Constants

The following constants are defined as properties on any objects implementing this subclause:

Name	Value	Use
COMPONENT_TYPE_VIDEO	0	Represents a video component. This constant is used for all video components regardless of encoding.
COMPONENT_TYPE_AUDIO	1	Represents an audio component. This constant is used for all audio components regardless of encoding.
COMPONENT_TYPE_SUBTITLE E	2	Represents a subtitle component. This constant is used for all subtitle components regardless of subtitle format. A subtitle component may also be related to closed captioning as part of a video stream.

7.16.5.2.2 Properties

function onSelectedComponentChanged(Integer componentType)	
This function is called when there is a change in the set of components being presented. This may occur if one of the currently selected components is no longer available and an alternative is chosen based on user preferences, or when presentation has changed due to a different component or set of components being selected. OITFs may optimise event dispatch by dispatching a single event in response to several calls to <code>selectComponent()</code> or <code>unselectComponent()</code> made in rapid succession. The specified function is called with one argument: • <code>Integer componentType</code> – The type of component whose presentation has changed, as represented by one of the constant values listed in 7.16.5.2.1. If more than one component type has changed, this argument will take the value <code>undefined</code>.	

7.16.5.2.3 Methods

AVComponentCollection getComponents(Integer componentType)		
Description	If the set of components is known, returns a collection of AVComponent values representing the components of the specified type in the current stream. If componentType is set to null or undefined, then all components are returned if they are known. For a video/broadcast object, the set of components shall be known if the video/broadcast object is in the presenting state and may be known if the object is in other states. For an A/V Control object, the set of components shall be known if the A/V Control object is in the playing state and may be known if the object is in other states. In the case of broadcast MPEG-2 transport streams, this method returns information from the PMT but the PMT is not always accurate. Components may be signalled in the PMT which are not actually present all the time. Components may be present but carrying information inconsistent with the PMT, for example a secondary audio stream may be signalled but carrying a copy of the primary audio stream when content for the secondary audio has not been produced. Applications can use the getSIDescriptors() method defined in 7.16.2.5 to obtain descriptors from the EIT where these subtleties are normally signalled. Exactly how they are "normally signalled" is generally market-specific. One or more of the components returned may be passed back to one of the other methods unchanged (e.g. selectComponent()). If property preferredAudioLanguage in the Configuration object (refer to 7.3.3) is set, then a component is by default selected and is considered as an active component. If property preferredSubtitleLanguage in the Configuration object (refer to 7.3.3) is set and property subtitleEnabled in AVOutput class (refer to 7.3.6.2) is enabled, then a component is by default selected and is considered as an active component.	
Arguments	componentType	The type of component to be returned , as represented by one of the constant values listed in 7.16.5.2.1.

AVComponentCollection getCurrentActiveComponents(Integer componentType)		
Description	If the set of components is known, returns a collection of <code>AVComponent</code> values representing the currently active components of the specified type that are being rendered. Otherwise returns <code>undefined</code>. For a video/broadcast object, the set of components shall be known if the video/broadcast object is in the presenting state and may be known if the object is in other states. For an A/V Control object, the set of components shall be known if the A/V Control object is in the playing state and may be known if the object is in other states. One or more of the components returned may be passed back to one of the other methods unchanged (e.g. <code>selectComponent()</code>).	
Arguments	<code>componentType</code>	The type of currently active component to be returned. represented by one of the constant values listed in 7.16.5.2.1.

void selectComponent(AVComponent component)		
Description	Select the component that will be subsequently rendered when A/V playback starts or select the component for rendering if A/V playback has already started. If playback has started, this shall replace any other components of the same type that are currently playing. If property preferredAudioLanguage in the Configuration object (refer to 7.3.3) is set, then a component is by default selected and it is not necessary to perform selectComponent(). If property preferredSubtitleLanguage in the Configuration object (refer to 7.3.3) is set and property subtitleEnabled in AVOutput class (refer to 7.3.6.2) is enabled, then a component is by default selected and it is not necessary to perform selectComponent().	
Arguments	component	A component object available in the stream currently being played.

void unselectComponent(AVComponent component)		
Description	Stop rendering of the specified component of the stream. If property preferredAudioLanguage in the Configuration object (see 7.3.3) is set, then unselecting a specific component returns to the default preferred audio language. If property preferredSubtitleLanguage in the Configuration object (see 7.3.3) is set and property subtitleEnabled in AVOutput class (see 7.3.6.2) is enabled, then unselecting a specific component returns to the default preferred subtitle language. In order to stop rendering subtitles completely it is necessary to disable subtitles with property subtitleEnabled in AVOutput class.	
Arguments	component	The component to be stopped.

void selectComponent(Integer componentType)		
Description	If A/V playback has already started, start rendering the default component of the specified type in the current stream. This shall replace any other components of the same type that are currently playing. If A/V playback has not started, the default component of the specified type will be subsequently rendered once playback does start.	
Arguments	componentType	The type of component for which the default component should be rendered.

void unselectComponent(Integer componentType)		
Description	If A/V playback has already started, stop rendering of the specified type of component. If A/V playback has not started, no components of the specified type will be subsequently rendered once playback does start.	
Arguments	componentType	The type of component to be stopped.

7.16.5.2.4 Events

For the intrinsic event "onSelectedComponentChange", corresponding DOM events shall be generated, in the following manner:

Intrinsic event	Corresponding DOM event	DOM Event properties
onSelectedComponentChange	SelectedComponentChange	Bubbles: No Cancellable: No Context Info: componentType

This DOM event is directly dispatched to the event target, and will not bubble nor capture. Applications should not rely on receiving these events during the bubbling or the capturing

phase. Applications that use DOM event handlers shall call the `addEventListener()` method on the `video/broadcast` object or AV Control object itself. The third parameter of `addEventListener`, i.e. "useCapture", will be ignored.

7.16.5.3 The AVComponent class

7.16.5.3.1 General

AVComponent represents a component within a complete media stream – a single stream of video, audio or data that can be played or manipulated. This is not necessary for basic playback, record or EPG services. However, it provides a mechanism to get at extended streams for enhanced services.

For forward compatibility the DAE application shall check the value of the `type` property to ensure that it is accessing an AVComponent object of the correct type.

7.16.5.3.2 Properties

readonly Integer componentTag
The component tag identifies a component. The component tag identifier corresponds to the <code>component_tag</code> in the component descriptor in the ES loop of the stream in the PMT ETSI EN 300 468, or undefined if the component is not carried in an MPEG-2 TS.

readonly Integer pid
The MPEG Program ID (PID) of the component in the MPEG2-TS in which it is carried, or undefined if the component is not carried in an MPEG-2 TS.

readonly Integer type
Type of the component stream. Valid values for this field are given by the constants listed in 7.16.5.2.1.

readonly String encoding										
The encoding of the stream. The value of video format or audio format defined in Clause 4 of IEC 62766-2-1:2016 shall be used. For subtitle components, the following values are used (all according to Clause 7 of IEC 62766-2-1:2016):										
<table><tr><th>Value</th><th>Description</th></tr><tr><td>DVB-SUBT</td><td>DVB subtitles</td></tr><tr><td>EBU-SUBT</td><td>EBU Teletext based subtitles</td></tr><tr><td>CEA-SUBT</td><td>CEA-708C Closed Captions</td></tr><tr><td>3GPP-TT</td><td>3GPP Timed Text</td></tr></table>	Value	Description	DVB-SUBT	DVB subtitles	EBU-SUBT	EBU Teletext based subtitles	CEA-SUBT	CEA-708C Closed Captions	3GPP-TT	3GPP Timed Text
Value	Description									
DVB-SUBT	DVB subtitles									
EBU-SUBT	EBU Teletext based subtitles									
CEA-SUBT	CEA-708C Closed Captions									
3GPP-TT	3GPP Timed Text									

readonly Boolean encrypted
Flag indicating whether the component is encrypted or not.

7.16.5.4 The AVVideoComponent class

7.16.5.4.1 General

The AVVideoComponent class implements the AVComponent interface.

7.16.5.4.2 Properties

readonly Number aspectRatio
Indicates the aspect ratio of the video or undefined if the aspect ratio is not known. Values shall be equal to width divided by height, rounded to a float value with two decimals, e.g. 1,78 to indicate 16:9 and 1,33 to indicate 4:3.

7.16.5.5 The AVAudioComponent class

7.16.5.5.1 General

The AVAudioComponent class implements the AVComponent interface.

7.16.5.5.2 Properties

readonly String language
An ISO 639-2 language code representing the language of the stream, as defined in ISO 639-2.

readonly Boolean audioDescription
Has value true if the stream contains an audio description intended for people with a visual impairment, false otherwise.

readonly Integer audioChannels
Indicates the number of channels present in this stream (e.g. 2 for stereo, 5 for 5.1, 7 for 7.1).

7.16.5.6 The AVSubtitleComponent class

7.16.5.6.1 General

The AVSubtitleComponent class implements the AVComponent interface.

7.16.5.6.2 Properties

readonly String language
An ISO 639-2 language code representing the language of the stream, as defined in ISO 639-2

readonly Boolean hearingImpaired
Has value true if the stream is intended for the hearing-impaired (e.g. contains a written description of the sound effects), false otherwise.

7.16.5.7 The AVComponentCollection class

```
typedef Collection<AVComponent> AVComponentCollection
```

An AVComponentCollection represents a collection of AVComponent objects. See Annex I for the definition of the collection template.

7.16.6 Additional support for protected content

The existing Download and Recording classes shall both be extended with two properties isEncrypted and DRMSystemIds and one method getDRMPrivateData as follows.

readonly Boolean isEncrypted
Has value true if the content is protected by a DRM. or undefined if this information is not available to the OITF. The mapping for different content formats is described in 8.4.4.

readonly StringCollection DRMSystemIDs
StringCollection object containing the names of the DRM system ID of the DRM protecting the content. OIPF DRMSystemID are defined in Table 9 of IEC 62766-3:2016. The collection is empty if this information is not available to the OITF. The mapping for different content formats is described in 8.4.4.

String getDRMPriateData(String DRMSystemID)		
Description	Gets the private opaque data in the protected content in the scope of the specified DRM system. The function returns the private data of the protected content in the scope of the DRM system. The function returns undefined if this information is not available to the OITF or if the content is not protected by the specified DRM. The mapping for different content formats is described in 8.4.4. The private data bytes shall be encoded in a string whose characters shall be restricted to the ISO Latin-1 character set. Each character in the string represents a byte of the private data, such that a byte at position "i" in the private data is equal the Latin-1 character code of the character at position "i" in the string.	
Arguments	DRMSystemID	DRM system ID as defined by element DRMSystemID in Table 9 of IEC 62766-3:2016.

7.17 DLNA RUI remote control function APIs

7.17.1 General

Subclause 7.17 defines the APIs related to the DLNA RUI RCF.

The DLNA RUI RCF APIs provide the necessary JavaScript properties and methods for a DAE application to communicate with remote control devices and provide a Control UI (i.e. one or more CE-HTML documents that enable the DAE application to be controlled from the remote control device) on such devices. Using these APIs, remote control devices can:

- obtain a Control UI from the OITF or the IPTV Applications server via the OITF;
- send information such as control messages to the OITF; and
- receive information from the OITF.

Subclause 7.17 shall apply for OITFs that have indicated **<remoteControlFunction>** with value "true" as defined in 9.3.17 in its capability description.

7.17.2 The application/oipfRemoteControlFunction embedded object

7.17.2.1 General

OITFs that have indicated **<remoteControlFunction>** with value "true" shall support the DLNA RUI RCF APIs through the use of the following non-visual embedded object:

```
<object type="application/oipfRemoteControlFunction"/>
```

7.17.2.2 Constants

The following constants are defined as properties of the application/oipfRemoteControlFunction embedded object:

Constant name	Numeric Value	Use
REQUEST_CUI	0	A remote control device (a Control UI or an XML UI Listing) requests a control UI by using the pre-defined URI <code>"/rcf/request_cui"</code> .
REQUEST_MSG	1	A Control UI in the remote control device sends a message by using the pre-defined URI <code>"/rcf/request_msg"</code> .
CREATE_APP	2	A Control UI in the remote control device sends a message by using a URI defined by an OITF. This message has triggered the application receiving this event to be launched by the OITF.

7.17.2.3 Properties

readonly Integer currentRemoteDeviceHandle
The handle of the remote control device that is associated with the DAE application in the mapping information table (see 8.5.7) and is waiting for the response from the DAE application. The value of this handle is assigned by the OITF, and is unique within the OITF for the duration of a session (the <i>duration</i> of the connection between the OITF and that remote control device). Applications shall not rely on the value of this handle being preserved across sessions. This property is retrieved from the mapping information table (see 8.5.7) in the OITF, which contains the pairing information between the remote control device and the DAE application. Only one remote control device is allowed to connect to a given DAE application at a time. If there is no mapping information between a remote control device and the DAE application; this property returns undefined.

readonly String currentRemoteDeviceUA
The remote control device User-Agent string that has been provided in the remote control device's HTTP request. The <code>application/oipfRemoteControlFunction</code> object stores the value of the User-Agent header included in the most recent HTTP request of the remote control device currently being connected to this DAE application. The User-Agent string of the remote control device is expected to conform to the format of the User-Agent string defined in [Req. 5.3.a] of CEA-2014-A. If there is no mapping information between a remote control device and the DAE application, this property returns undefined.

<pre>function onReceiveRemoteMessage(Integer requestType, Integer remoteDeviceHandle, Integer reqHandle, String requestLine, String headers, String body)</pre>
The function that is called when the remote control device sends an HTTP request with one of the pre-defined URIs ("/rcf/request_cui" or "/rcf/request_msg"), or sends an HTTP request to the OITF to launch a DAE application. The DAE application can distinguish between these two cases by the type parameter as follows: - When the remote control device requests a control UI by using the pre-defined URI "/rcf/request_cui", the function is called with the type parameter REQUEST_CUI. - When the remote control device sends a message by using the pre-defined URI "/rcf/request_msg", the function is called with the type parameter REQUEST_MSG. When the DAE application is launched by the OITF in response to a request from the control UI in the remote control device, the function is called with the type parameter CREATE_APP. The function will be called after the DAE application has loaded (i.e. after the onLoad event has been dispatched to the DAE application). The DAE application being launched is expected to contain an instance of the application/oipfRemoteControlFunction object. The OITF shall dispatch the event to the application/oipfRemoteControlFunction object in the DAE application matched with the remote control device handle which are paired in the mapping information table (see 8.5.7). When the ReceiveRemoteMessage event is dispatched to the target application, the DAE application receives the remote control device's User-Agent header value containing the remote control device's capability (in the headers parameter) which the OITF was given with the HTTP request from the remote control device. The DAE application shall include the User-Agent value from the remote control device in the XMLHttpRequest object it uses to retrieve the appropriate Control UI from the IPTV Applications server (see 8.1.3). When this event is invoked, the DAE application shall respond by calling the sendRemoteMessage() method. This method need not be called from the event handling function, and may be called after a request to the IPTV Applications Server for an appropriate Control UI has completed. Only one remote control device is allowed to connect to a DAE application (see 8.5.7) at any time. If an HTTP request from another remote control device directed at the DAE application is received by the OITF while a remote control device is connected, the OITF shall not make and dispatch ReceiveRemoteMessage events to the target DAE application but shall send an HTTP response (HTTP 500 – Internal Server Error) to the remote control device. Every HTTP request from a remote control device to the DAE application with which it is paired shall generate an onReceiveRemoteMessage event, even if there are previous HTTP requests which the DAE application has not yet responded to. Each HTTP request shall be given a unique reqHandle by the OITF to allow the DAE application to distinguish between outstanding requests. The specified function is called with six arguments: Type, remoteDeviceHandle, reqHandle, requestLine, headers and body which are defined as follows: Integer requestType – the type of the HTTP request from the remote control device. This shall take one of the following values: - REQUEST_CUI - REQUEST_MSG - CREATE_APP Integer remoteDeviceHandle – the handle of the remote control device that is sending the HTTP request to the DAE application. This handle has a unique value which is assigned by the OITF. Integer reqHandle – the handle of the request from the remote control device. The value of this handle is assigned by the OITF, and is unique within the OITF for the duration of a session (the duration of the connection between the OITF and that remote control device). Applications shall not rely on the value of this handle being preserved across sessions. String requestLine – the HTTP requestLine string that comes from the remote control device.
String headers – the HTTP request header string that comes from the remote control device. String body – the HTTP request body that comes from the remote control device. The values of the requestLine, headers and body parameters are derived from the received HTTP request as follows: Where: HTTP Request = Request-Line CRLF Header-Lines CRLF Message Header-Lines = *((general-header request-header entity-header) CRLF) Message = [message-body] Then: requestLine = "Request-Line". headers = "Header-Lines". body = "Message".

function onResultMulticastNotif(Integer remoteDeviceHandle, Integer reqHandle, Boolean dynamic)
The function that is called when the remote control device sends an HTTP request with an URL which is a value of a <ruieventURL> element in the Multicast Notification Message. This function is called with the following arguments: - Integer remoteDeviceHandle – the handle of the remote control device which is sending the HTTP request to the DAE application. - Integer reqHandle – the handle of the request from the remote control device. - Boolean dynamic – if true, the DAE application shall respond by calling the sendRemoteMessage method. This method need not be called from the event handling function, and may be called after a request to the IPTV Applications Server for an appropriate notification CE-HTML document has completed.

7.17.2.4 Methods

Boolean useServerSideXMLUIListing(String xmlUIListing, boolean advertiseImmediately)		
Description	Generate an XML UI Listing by merging the XML UI Listing currently being exposed by the DLNA RUIS in the OITF with the XML UI Listing provided by the xmlUIListing parameter of this method. If the OITF successfully generates the new XML UI Listing, this method shall return true. Otherwise, it shall return false.	
Arguments	xmlUIListing	The Server Side XML UI Listing.
	advertiseImmediately	After generating the new XML UI Listing, if this parameter is true, the DLNA RUIS in the OITF shall send a UPnP Discovery (SSDP:byebye) message followed by a UPnP Discovery (SSDP:alive) message. This notifies the DLNA RUC in any remote control device that it should retrieve the new XML UI Listing.

Boolean sendRemoteMessage(Integer remoteDeviceHandle, Integer reqHandle, String headers, String message)		
Description	Send the HTTP response with the headers and the message to the remote control device related to remoteDeviceHandle. This method is called by a DAE application in response to a HTTP request from the remote control device. This method can be called at any time for any pending HTTP request (i.e. a request with handle reqHandle from the remote control device with handle remoteDeviceHandle that has not had a response from the OITF via a sendRemoteMessage() or sendInternalServerError() call). This method shall return true if the operation succeeded, or false if failed. If there is no HTTP connection, it also returns false.	
Arguments	remoteDeviceHandle	The handle of the remote control device.
	reqHandle	The handle of the request as provided by onReceiveRemoteMessage .
	headers	The HTTP response header string. This string is added to the default HTTP header string generated by the OITF to form the HTTP header string used for the HTTP response. Any parameters that are specified in both strings shall be set to the value in the headers argument. If the headers supplied by the application do not include a Content-Type header, the OITF shall use the default content type of application/ce-html+xml .
	message	The HTTP response body string whose type is text (e.g. XML, JSON, CE-HTML or Plain Text).

Boolean sendMulticastNotif(Integer remoteDeviceHandle, Integer eventLevel, String notifCEHTML, String friendlyName, String profileList)														
Description	Send the 3rd party multicast notification to any remote control devices (as defined in 5.6.1 of CEA-2014-A) based on target remote Device information. The OITF shall store the text (essentially a CE-HTML document) provided in the notifCEHTML parameter inside the DLNA RUIS and shall create a URL to it which can be used by remote control devices to retrieve the original text. This URL shall be inserted in the <uiEventURL> element in the Multicast Notification Message. If the notifCEHTML parameter is set to null, the HTTP request from the remote Device to retrieve the text shall be being pended and dispatch the onResultMuticastNotif event to the DAE application, which will retrieve a CE-HTML document dynamically. The DAE application shall use the sendRemoteMessage method with a CE-HTML document related parameters to send the text (notification message). If the remoteDeviceHandle parameter in this method has a value other than -1, the notification CE-HTML document will be retrieved by the only remote Device matched with the remoteDeviceHandle parameter, whereas if the parameter has -1, all the remote Devices could retrieve the notification CE-HTML document from the OITF (see 8.5.6). This method shall return true if the operation succeeded, or false if it failed.													
Arguments	remoteDeviceHandle	The handle of the remote Device.												
	eventLevel	The value of the HTTP LVL. This allows the remote control devices to filter the multicast notification messages. The following are the defined event levels and the expected meaning of those values (see 5.6.1 of CEA-2014-A for more information): <table><tr><th>Status</th><th>Semantics</th></tr><tr><td>0</td><td> The "urn:upnp:/emergency" is included in the LVL header of the multicast notification. The event carries critical information that the remote control device should act upon immediately. </td></tr><tr><td>1</td><td> The "urn:upnp:/fault" is included in the LVL header of the multicast notification. The event carries information related to an error case. </td></tr><tr><td>2</td><td> The "urn:upnp:/warning" is included in the LVL header of the multicast notification. The event carries information that is a non-critical condition that the remote control device may want to process or pass to the user. </td></tr><tr><td>3</td><td> The "urn:upnp:/info" is included in the LVL header of the multicast notification. The event caries informational contents that is not part of the main service interaction but may be useful to some remote control devices in some circumstances, such as debugging information or other data. </td></tr><tr><td>4</td><td> The "urn:upnp:/general" is included in the LVL header of the multicast notification. For events that fit into no other defined category. </td></tr></table>	Status	Semantics	0	The "urn:upnp:/emergency" is included in the LVL header of the multicast notification. The event carries critical information that the remote control device should act upon immediately.	1	The "urn:upnp:/fault" is included in the LVL header of the multicast notification. The event carries information related to an error case.	2	The "urn:upnp:/warning" is included in the LVL header of the multicast notification. The event carries information that is a non-critical condition that the remote control device may want to process or pass to the user.	3	The "urn:upnp:/info" is included in the LVL header of the multicast notification. The event caries informational contents that is not part of the main service interaction but may be useful to some remote control devices in some circumstances, such as debugging information or other data.	4	The "urn:upnp:/general" is included in the LVL header of the multicast notification. For events that fit into no other defined category.
	Status	Semantics												
	0	The "urn:upnp:/emergency" is included in the LVL header of the multicast notification. The event carries critical information that the remote control device should act upon immediately.												
	1	The "urn:upnp:/fault" is included in the LVL header of the multicast notification. The event carries information related to an error case.												
2	The "urn:upnp:/warning" is included in the LVL header of the multicast notification. The event carries information that is a non-critical condition that the remote control device may want to process or pass to the user.													
3	The "urn:upnp:/info" is included in the LVL header of the multicast notification. The event caries informational contents that is not part of the main service interaction but may be useful to some remote control devices in some circumstances, such as debugging information or other data.													
4	The "urn:upnp:/general" is included in the LVL header of the multicast notification. For events that fit into no other defined category.													
notifCEHTML	The text that makes up the notification CE-HTML document, the link to which is sent to the remote control device.													
profileList	All the profiles that the remote UI Server in the OITF requires the remote UI client in the remote control device to support to properly render the notification CE-HTML document. The value of the <profilelist> element shall conform to the definition of the <profilelist> element in the XML schema in Annex B of CEA-2014-A.													

Boolean sendInternalServerError (Integer remoteDeviceHandle, Integer reqHandle)		
Description	Send the HTTP status code (500: Internal Server Error) in response to a pending HTTP request from the remote control device. This method shall return true if the operation succeeded, or false if it failed.	
Arguments	remoteDeviceHandle	The handle of the remote control device.
	reqHandle	The handle of the request as provided by onReceiveRemoteMessage .

Boolean dropConnection (Integer remoteDeviceHandle)		
Description	Remove the mapping information in the table between the DAE application and the remote control device currently bound to the DAE application. This method shall return true if the operation succeeded, or false if it failed.	
Arguments	remoteDeviceHandle	The handle of the remote control device.

7.17.2.5 Events

For the intrinsic events listed in the table below, a corresponding DOM event shall be generated in the following manner:

Intrinsic event	Corresponding DOM event	DOM Event properties
onReceiveRemoteMessage	ReceiveRemoteMessage	Bubbles: No Cancellable: No Context Info: requestType, remoteDeviceHandle, reqHandle, requestLine, headers, body
onResultMuticastNotif	ResultMuticastNotif	Bubbles: No Cancellable: No Context Info: remoteDeviceHandle, reqHandle, dynamic

The above DOM events are directly dispatched to the event target, and will not bubble nor capture. Applications should not rely on receiving **ReceiveRemoteMessage** or a **ResultMuticastNotif** event during the bubbling or the capturing phase. Applications that use DOM event handlers shall call the **addEventListener()** method on the application/objectRemoteControlFunction object. The third parameter of **addEventListener**, i.e. "useCapture", will be ignored.

8 System integration aspects

8.1 HTTP protocol

8.1.1 General

In addition to what is required by 5.3 of CEA-2014-A, an OITF shall apply the following requirements.

8.1.2 HTTP User-Agent header

All DAE application HTTP requests shall include a User-Agent header using the syntax described in this subclause. Embedded objects HTTP requests may include a User-Agent header using this syntax.

The User-Agent header shall include:

```
OIPF-<oipfProfile>/<releaseVersion>.<majorVersion>.<minorVersion>
(<capabilities>; [<vendorName>]; [<modelName>]; [<softwareVersion>];
[<hardwareVersion>]; [<familyName>]; <reserved>) [<appName>[/<appversion>]]
```

Where:

- the <capabilities> field consists of a description of the OITFs capabilities. Valid values include:
 - a base profile string concatenated with one or more optional profile name fragment strings, such as the base UI profile strings and UI profile name fragment strings as defined in 9.2;
- the <oipfProfile> field identifies the profile implemented by the OITF as defined in the specification of the oipfProfile property of the LocalSystem class (in 7.3.4);
- the <releaseVersion>, <majorVersion> and <minorVersion> fields identify the version of the specification implemented by the OITF as defined in 7.3.4 with properties of the same name;
- the <vendorName>, <modelName>, <familyName>, <softwareVersion> and <hardwareVersion> fields are the same as the one defined in 7.1.2 and are optional;
- the <reserved> field is reserved for future extensions;
- the <appName> and <appVersion> fields are defined in the window.navigator object and are optional.

This User-Agent header may be extended with other implementation-specific information.

Valid examples of such syntax are:

```
User-Agent: OIPF-OIP/2.3.0 (OITF_HD_UIPROF+PVR+DL; Sonic; TV44; 1.32.455; 2.002;
com.acme.2012;) Bee/3.5
User-Agent: OIPF-BMP/2.3.0 (OITF_HD_UIPROF+PVR+DL;;;;;)
```

8.1.3 HTTP X-OITF-RCF-User-Agent header

When the DAE application or embedded object ("application/oipfRemoteControlFunction") makes an HTTP request for the Control UI to the IPTV Applications server, the value of the X-OITF-RCF-User-Agent header shall be filled with the value of the User-Agent header provided by the DAE application (and which came from the DLNA RUC on the remote control device).

8.2 Mapping from APIs to protocols

8.2.1 General

Subclause 8.2 describes mapping of DAE APIs to the specific protocol entities as defined in the protocol specification IEC 62766-4-1.

Subclause 8.2.2 describes mappings that apply to CoD download over HTTP.

Subclause 8.2.3 describes mappings that apply to CoD unicast streaming with SIP session management.

Subclause 8.2.4 describes mappings that apply to Multicast Streaming of Scheduled Content with SIP session management.

Subclause 8.2.5 describes mappings that apply to Communication Services with SIP session management.

Subclause 8.2.6 describes mappings that apply to CoD unicast streaming over RTP and HTTP.

Subclause 8.2.7 describes mappings that apply to Scheduled Content Multicast Streaming.

Subclause 8.2 provides details of mapping of the DAE APIs to the descriptions provided in the Protocol specification for APIs between the OITF and the Network over reference points UNIT-17.

8.2.2 CoD download over HTTP

This subclause provides details of mapping of the DAE APIs to the descriptions provided in the Protocol specification for APIs between the OITF and the Network over reference points UNIT-17 for download over HTTP.

Methods	Procedures
<code>registerDownload(String contentAccessDownloadDescriptor, Date downloadStart, Integer priority)</code>	API described in 7.4.2.2 to download content described in the <code>contentAccessDownloadDescriptor</code>. Data structure of the <code>contentAccessDownloadDescriptor</code> as described in Clause C.1. If the OITF includes the Content Download functional entity, the information in the <code>contentAccessDescriptor</code> is passed to the Content Download functional entity to download content over UNIT-17 using HTTP as described in 6.3.4.2 of IEC 62766-4-1:2017 and 4.7.5.
<code>registerDownload(String URL, String contentType, Date downloadStart, Integer priority)</code>	API described in 7.4.2.2 to download the content identified by the given URL. If the OITF includes the Content Download functional entity, the URL is passed to the Content Download functional entity to download content over UNIT-17 using HTTP as described in 6.3.4.2 of IEC 62766-4-1:2017. As specified in 7.4.2.2, the <code>contentType</code> attribute can be used to evaluate if the content type is part of the list of accepted content types of the OITF. If <code>contentType</code> has value "application/vnd.oipf.ContentAccessDownload+xml", the method shall return a download identifier, after which the OITF shall immediately fetch the content access download descriptor, after which the same shall happen as if <code>registerDownload()</code> had been called.
<code>registerDownloadFromCRID(String CRID, String IMI, Date downloadStart, Integer priority)</code>	API described in 7.4.3 to download content described in a BCG record. If the OITF includes the Content Download functional entity, <CRID,IMI> BCG tuple is resolved to an URL as described in 6.3 of IEC 62766-4-1:2017 and passed to the Content Download functional entity to download content over UNIT-17 using HTTP as described in 6.3.4.2 of IEC 62766-4-1:2017.

8.2.3 CoD unicast streaming with SIP session management

This subclause provides details of mapping of the DAE APIs to the descriptions provided in the Protocol specification IEC 62766-4-1 for APIs between the OITF and the Network over reference points HNI-IGI, UNIS-11 and UNIT-17U for CoD Unicast Streaming with SIP session management.

Method	Procedures												
play(Number speed)	Selection of a content item results in session initiation and access to content stream. Parameters needed to build the offer SDP may be pre defined locally in the OITF or the OITF shall request the IG to retrieve missing SDP parameters as described in 6.3.2.1 of IEC 62766-4-1:2017. If the OITF does not have all transport parameters (RTP or UDP transport for MPEG2TS encapsulation or direct RTP, FEC layers addresses and ports), code information or bandwidth information to populate the SDP the OITF shall prompt the IG to send OPTIONS request in order to retrieve the missing parameters. The OITF shall provide the following information for the OPTIONS request. Not all required headers are listed. Refer to the Protocol specification IEC 62766-4-1 for a complete list. <table border="1"> <tr> <td>X-OITF-Request-Line</td><td>Identify the HNI-IGI method with the content identifier as described by the data property, e.g. OPTION sip:PSI-Twister@IPTV_Service_Control.orange.com SIP/2.0</td></tr> <tr> <td>X-OITF-From</td><td>Local defined OITF CurrentUser property, e.g. <sip:family@ims.live.ericsson.com>; tag=1211455936632545012</td></tr> <tr> <td>X-OITF-To</td><td>Copied from the data property, e.g. sip: PSI-Twister@IPTV_Service_Control.orange.com</td></tr> </table> The response to the OPTIONS message request contains the information to populate the SDP offer. The OITF prepares an SDP offer and requests the IG to initiate a session, in addition to the SDP the following parameters are forwarded from the OITF to the IG. Not all required headers are listed. Refer to the Protocol specification IEC 62766-4-1 for a complete list. <table border="1"> <tr> <td>X-OITF-Request-Line</td><td>Identify the HNI-IGI method with the content identifier as described by the data property, e.g. INVITE sip:PSI-Twister@IPTV_Service_Control.orange.com SIP/2.0</td></tr> <tr> <td>X-OITF-From</td><td>Local defined OITF CurrentUser property, e.g. <sip:family@ims.live.ericsson.com>; tag=1211455936632545012</td></tr> <tr> <td>X-OITF-To</td><td>Copied from the data property, e.g. sip: PSI-Twister@IPTV_Service_Control.orange.com</td></tr> </table> After a successful session setup the OITF shall use the media player to access the RTSP URI with the session ID negotiated and received as part of the SDP offer, described in 8.2.1.2 of IEC 62766-4-1:2017. The OITF shall send an RTSP PLAY over UNIS-11 using attribute values received in the SDP from the session initiation procedure. The RTSP PLAY is as described in 8.2.1.2 of IEC 62766-4-1:2017.	X-OITF-Request-Line	Identify the HNI-IGI method with the content identifier as described by the data property, e.g. OPTION sip:PSI-Twister@IPTV_Service_Control.orange.com SIP/2.0	X-OITF-From	Local defined OITF CurrentUser property, e.g. <sip:family@ims.live.ericsson.com>; tag=1211455936632545012	X-OITF-To	Copied from the data property, e.g. sip: PSI-Twister@IPTV_Service_Control.orange.com	X-OITF-Request-Line	Identify the HNI-IGI method with the content identifier as described by the data property, e.g. INVITE sip:PSI-Twister@IPTV_Service_Control.orange.com SIP/2.0	X-OITF-From	Local defined OITF CurrentUser property, e.g. <sip:family@ims.live.ericsson.com>; tag=1211455936632545012	X-OITF-To	Copied from the data property, e.g. sip: PSI-Twister@IPTV_Service_Control.orange.com
X-OITF-Request-Line	Identify the HNI-IGI method with the content identifier as described by the data property, e.g. OPTION sip:PSI-Twister@IPTV_Service_Control.orange.com SIP/2.0												
X-OITF-From	Local defined OITF CurrentUser property, e.g. <sip:family@ims.live.ericsson.com>; tag=1211455936632545012												
X-OITF-To	Copied from the data property, e.g. sip: PSI-Twister@IPTV_Service_Control.orange.com												
X-OITF-Request-Line	Identify the HNI-IGI method with the content identifier as described by the data property, e.g. INVITE sip:PSI-Twister@IPTV_Service_Control.orange.com SIP/2.0												
X-OITF-From	Local defined OITF CurrentUser property, e.g. <sip:family@ims.live.ericsson.com>; tag=1211455936632545012												
X-OITF-To	Copied from the data property, e.g. sip: PSI-Twister@IPTV_Service_Control.orange.com												

Method	Procedures						
	The RTSP fields in the RTSP PLAY message shall be filled as follows: - The RTSP URL shall be set from the SDP h-uri attribute in the case of an absolute URI. The "data" property shall be updated with the SDP h-uri attribute. If the value of h-uri is a relative URI that is in the form of a media path, then the RTSP absolute URL is constructed by the OITF using the SDP IPAddress (from c-line) and port (from m-line) as the base followed by h-uri value for the media path. (e.g. rtsp://10.5.1.72:22554/TV3/823527) The RTSP Scale header shall be set to the value specified in argument speed in method play. The argument should equal one of the values in the playSpeeds property. The Scale values IETF RFC 2326, 12.34 are as follows: 1 indicates normal play; - if not 1, the value corresponds to the rate with respect to normal viewing rate; - a negative value indicates reverse direction. If the speed argument of method play does not equal a supported play speed indicated by the playSpeeds property, the player shall play the content at the closest available playback speed. The play() method should only return false if the best effort to play back the file at any speed has failed. The actual playback speed shall be available through the speed property of the A/V Control object. After a successful response to the RTSP PLAY message has been received, the OITF shall generate a PlaySpeedChanged event indicating the actual playback speed.						
stop()	The method enables the OITF to terminate an ongoing CoD session. The OITF shall request the IG to terminate the session as described in IEC 62766-4-1:2017, 6.3.2.1. The OITF shall include the following information from the request. Not all required headers are listed. Refer to the Protocol specification IEC 62766-4-1:2017, for a complete list. <table border="1"> <tr> <td>X-OITF-Request-Line</td><td>Identify the HNI-IGI method with the content identifier as described by the data property.eg. BYE sip:PSI-Twister@IPTV_Service_Control.orange.com SIP/2.0</td></tr> <tr> <td>X-OITF-From</td><td>Local defined OITF CurrentUser property. e.g. <sip:family@ims.live.ericsson.com>; tag=1211455936632545012</td></tr> <tr> <td>X-OITF-To</td><td>Copied from the data property. e.g. sip: PSI-Twister@IPTV_Service_Control.orange.com</td></tr> </table> The OITF shall remove all context information relevant to the terminated COD session upon a successful response from the IG.	X-OITF-Request-Line	Identify the HNI-IGI method with the content identifier as described by the data property.eg. BYE sip:PSI-Twister@IPTV_Service_Control.orange.com SIP/2.0	X-OITF-From	Local defined OITF CurrentUser property. e.g. <sip:family@ims.live.ericsson.com>; tag=1211455936632545012	X-OITF-To	Copied from the data property. e.g. sip: PSI-Twister@IPTV_Service_Control.orange.com
X-OITF-Request-Line	Identify the HNI-IGI method with the content identifier as described by the data property.eg. BYE sip:PSI-Twister@IPTV_Service_Control.orange.com SIP/2.0						
X-OITF-From	Local defined OITF CurrentUser property. e.g. <sip:family@ims.live.ericsson.com>; tag=1211455936632545012						
X-OITF-To	Copied from the data property. e.g. sip: PSI-Twister@IPTV_Service_Control.orange.com						
seek(Integer pos)	If the seek() method is called while the player is in the "playing" state, it sets current play position to "pos", by using the "Range" parameter in the RTSP PLAY as described in 8.2.1.2 of IEC 62766-4-1:2017. After a successful response to the RTSP PLAY message has been received, the OITF shall generate a PlayPositionChanged event indicating a new playback position of "pos". If the seek() method is called while the player is in the "paused" state, the value of playPosition is changed to reflect the new play position. This is the new play position that shall be used for the "Range" parameter of the RTSP PLAY message when playback is resumed.						

Method	Procedures
<code>play()</code>	This method causes the OITF to send an RTSP PAUSE message (refer to 7.1.1.2 of IEC 62766-4-1:2017). The RTSP PAUSE message shall include: - the RTSP URL shall be set to the value retrieved from the <code>fmtp:iptv_rtsp h-uri</code> attribute of the SDP answer; - session header shall be set as specified in the SDP answer <code>fmtp:iptv_rtsp h-session</code> attribute. After a successful response to the RTSP PAUSE message has been received, the OITF shall generate a <code>PlaySpeedChanged</code> event indicating a playback speed of 0.
<code>next()</code>	Not Supported. Note that track information is not supported in the protocol specification and therefore out of scope.
<code>previous()</code>	Not Supported. Note that track information is not supported in the protocol specification and therefore out of scope.

Property	Procedures
<code>read/write String data</code>	This property holds the URL that identifies the content, as defined in 4.8.2. See 7.3.2.1.1 of IEC 62766-4-1:2017 for details on CoD URI. It is used by the OITF compose the following headers for requests towards the IG X-OITF-Request-Line X-OITF-To If the "data" property of the A/V Control object refers to a Content-Access Streaming Descriptor (i.e. the object has type <code>"application/vnd.oipf.ContentAccessStreaming+xml"</code> as defined in 7.14.2.6), the OITF shall perform the following steps prior to performing the procedures defined in IEC 62766-4-1 as described for method <code>play()</code>: An HTTP GET request shall be made with the Request-URI set to the URL of the Content-Access Descriptor as denoted by the "data" property of the A/V Control object. After the server has returned a Content Access Streaming Descriptor (i.e. a document with type <code>"application/vnd.oipf.ContentAccessStreaming+xml"</code>), the OITF shall interpret the contents of the Content-Access Descriptor and choose a URL defined by one of the <code><ContentURL></code> elements. The criteria for choosing a URL can be the DRM system supported by the OITF. The URL shall then be used for setting up a Streaming CoD session, after which playback can be started (when the <code>play()</code> method is invoked). The "data" property of the AV object shall be changed to represent the chosen URL. Based on the information retrieved from the Content-Access Streaming Descriptor, the OITF shall passing the <code><DRMControlInformation></code> to the appropriate DRM agent, and should initialize the AV playback, i.e. by loading the correct codecs as identified by the Content-access Streaming Descriptor.
<code>readonly Number playPosition</code>	The property holds the current play position in milliseconds of the media referenced by the data property. The property value shall be based on the value retrieved using the RTSP GET_PARAMETERS method and parameter "position" (refer to 8.2.1.2 of IEC 62766-4-1:2017) adjusted for played duration and used scale. If information is not available the value shall be <code>undefined</code>. Note this may happen at the beginning of playing a video and GET_PARAMETER has not returned a value.

Property	Procedures
readonly Number playSpeeds[]	The property holds the available speeds, or referred in RTSP as Scale, to be used to change the playback speed. The property value shall be based on the value retrieved using RTSP GET_PARAMETERS method and parameter "scales" (refer to 7.1.1.2 of IEC 62766-4-1:2017). If information is not available the value shall be undefined . Note this may happen at the beginning of playing a video and GET_PARAMETER has not returned a value.
readonly Number playTime	The property holds the total duration in milliseconds of the media referenced by the data property. The property value shall be based on the value retrieved using RTSP GET_PARAMETER method and parameter "duration" (refer to 7.1.1.2 of IEC 62766-4-1:2017). If information is not available, the value shall be undefined . Note that this may happen at the beginning of playing a video and GET_PARAMETER has not returned a value.
readonly Number playState	No procedures defined since it is not related to protocol specification.
readonly Number error	No procedures defined since it is not related to protocol specification.
readonly Number speed	Float value indicating the actual playback speed for the content referenced by the data property. The normal default playback speed is represented by value 1.

Intrinsic event	Procedure
onPlaySpeedChanged	When RTSP ANNOUNCE with either beginning-of-stream or end-of-stream codes arrives the OITF shall generate onPlaySpeedChanged event with a speed value of 0.
onPlayPositionChanged	When the response to the RTSP PLAY with Range header request (Range is included when performing seek() with a position) the OITF shall generate onPlayPositionChanged event with the accepted position.

8.2.4 Scheduled content multicast streaming with SIP session management

8.2.4.1 General

Subclause 8.2.4 provides details of mapping of the DAE APIs to the descriptions provided in the Protocol specification IEC 62766-4-1 for APIs between the OITF and the Network over reference points HNI-IGI, UNIS-11, UNIS-13 and UNIT 17 for Scheduled Content multicast streaming with SIP session management.

8.2.4.2 Conveyance of channel list

Service discovery description procedure as described in 7.2.4.1 of IEC 62766-4-1:2017, enables the OITF to obtain the URL to access the broadcast channel information. The OITF shall use UNIS-7 using this URL to obtain the Broadcast Discovery Record.

8.2.4.3 Switching channels

Methods	Procedures						
<code>setChannel(Channel channel, Boolean trickplay, String contentAccessDescriptorURL)</code>	The <code>setChannel()</code> method of the <code>video/broadcast</code> object shall be used to initiate a broadcast session or switch channels. The procedures that are performed over the HNI-IGI reference point depend on the current state of broadcast session, either it is active or not. Note that an inactive broadcast session means that no service is being viewed. If the channel is an IMS based IPTV service (i.e., if it is of type <code>ID_IPTV_SDS</code> and if the corresponding service has a "sip-igmp-rtp-udp" or "sip-igmp-udp" file format specified in its SD&S BDR record), the following steps are taken: Session Initiation The OITF shall generate a session initiation request over the HNI-IGI including and SDP offer as described in 6.3.2 of IEC 62766-4-1:2017. The bandwidth is set according to the explanation under heading "Selection of Bandwidth" further down. If a "contentAccessDescriptorURL" has been specified for the <code>setChannel()</code> method, the OITF shall perform the following steps prior to performing the procedures defined in IEC 62766-4-1 for performing <code>setChannel()</code> as described below: • a HTTP GET request shall be made with the Request-URI set to the URL of the Content-Access Descriptor as denoted by the "contentAccessDescriptor" attribute; • based on the information retrieved from the Content-Access Descriptor, the OITF shall passing the <code><DRMControlInformation></code> to the appropriate DRM agent. The OITF shall provide the following information as part of the scheduled session initiation request as described in 7.3.3.1 of IEC 62766-4-1:2017. Not all required headers are listed. Refer to the Protocol specification IEC 62766-4-1 for a complete list. <table border="1"> <tr> <td>X-OITF-Request-Line</td><td>Identify the HNI-IGI method with the well known PSI Service Identifier) of the scheduled content, e.g. INVITE sip:IPTV_SC_Service@iptv.ericsson. SIP/2.0</td></tr> <tr> <td>X-OITF-From</td><td>Local defined OITF <code>CurrentUser</code> property, e.g. <sip:family@ims.live.ericsson.com> tag=1211455936632545012</td></tr> <tr> <td>X-OITF-To</td><td>PSI of the scheduled content, e.g. sip:IPTV_SC_Service@iptv.ericsson.</td></tr> </table> The Offer SDP included in the OITF shall have attributes as described in Clause C.2 of IEC 62766-4-1:2017. On positive response to the INVITE request the OITF shall send an IGMP Join request on the UNIS-13 as described in 9.2.1.1 of IEC 62766-4-1:2017. Session Modification If the bandwidth conditions change as described under heading "Selection of Bandwidth" further down, then the OITF shall generates a session modification request over the HNI-IGI including the new SDP offer. The OITF shall provide the following information as part of the scheduled session modification request as described in 7.3.3.1 of IEC 62766-4-1:2017. Not all required headers are listed. Refer to the Protocol specification IEC 62766-4-1 for a complete list.	X-OITF-Request-Line	Identify the HNI-IGI method with the well known PSI Service Identifier) of the scheduled content, e.g. INVITE sip:IPTV_SC_Service@iptv.ericsson. SIP/2.0	X-OITF-From	Local defined OITF <code>CurrentUser</code> property, e.g. <sip:family@ims.live.ericsson.com> tag=1211455936632545012	X-OITF-To	PSI of the scheduled content, e.g. sip:IPTV_SC_Service@iptv.ericsson.
X-OITF-Request-Line	Identify the HNI-IGI method with the well known PSI Service Identifier) of the scheduled content, e.g. INVITE sip:IPTV_SC_Service@iptv.ericsson. SIP/2.0						
X-OITF-From	Local defined OITF <code>CurrentUser</code> property, e.g. <sip:family@ims.live.ericsson.com> tag=1211455936632545012						
X-OITF-To	PSI of the scheduled content, e.g. sip:IPTV_SC_Service@iptv.ericsson.						

Methods	Procedures	
	X-OITF-Request-Line Identify the HNI-IGI method with the well known PSI (Public Service Identifier) of the scheduled content. e.g. INVITE sip:IptvBroadcast@iptv.ericsson.com SIP/2.0	
	X-OITF-From Local defined OITF CurrentUser property. e.g. <sip:family@ims.live.ericsson.com>; tag=1211455936632545012	
	X-OITF-To PSI of the scheduled content. e.g. sip:IptvBroadcast@iptv.ericsson.com	
	The Offer SDP included by the OITF shall have attributes as relevant to the new channel as described in Clause C.2 of IEC 62766-4-1:2017. On receiving a successful response to the INVITE request the OITF shall send and IGMP Leave and IGMP Join request on the UNIS-13 as described in 8.1.1.1 of IEC 62766-4-1:2017. No Session Modification If the bandwidth conditions as described under heading "Selection of Bandwidth" further down have not changed, then the OITF shall send a membership report to leave the previously viewed channel, if applicable, and with the same membership report join to the multicast group associated with the selected channel. The multicast group information is retrieved from the Broadcast Discovery Record. Selection of Bandwidth The bandwidth to be used for the broadcast session depends on the information provided in the Broadcast Discovery Record (refer to 4.3.3.2 of IEC 62766-3:2016). The Broadcast Discovery Record uses the term "service" to indicate a channel. If the TimeToRenegotiate (TTR) element is not provided within the IPService of the Broadcast Discovery Record then the bandwidth shall be based on the maximum bandwidth for all the services in the Broadcast Discovery Record. In this case only one session initiation is performed at initial activation of broadcast service, and no session modification is required. If the TTR element is provided then the MaxBitrate from the new service and current service are compared. If broadcast service is not active and there is no active current service, session initiation is performed with the new service MaxBitrate. For already active broadcast service there are three conditions. • If the MaxBitrate of the new service is greater than that of the current service and the reserved bandwidth is exceeded, network bandwidth reservation using the MaxBitrate of the new service shall occur immediately with session modification to ensure sufficient bandwidth is made available for the new service. • If the MaxBitrate of the new service is equal to that of the current service, network bandwidth reservation procedures shall not be performed as sufficient bandwidth is already available for the new service. • If the MaxBitrate of the new service is less than that of the current service and there is no pending TTR timer, a timer using the TTR element of the new service is started, which will renegotiate the bandwidth with session modification. Note that at every channel change, if there is a pending timeout for session modification due to a previous service change, then the timer is restarted. When the timer expires, the bandwidth for the currently viewed service is used in a session modification. The session initiation, session modification and no session modification are further described above.	

8.2.4.4 End broadcast service

Methods	Procedures						
release()	The release method of the video/broadcast object causes the OITF to perform an IGMP Leave on the active broadcast session as described in 9.2.1.1 of IEC 62766-4-1:2017. If the channel has an <code>idType</code> of <code>ID_IPTV_SDS</code>, the OITF shall then execute a session termination procedure by sending a BYE request over the HNI-IGI interface as described in 6.3.1.2 of IEC 62766-4-1:2017. The request shall include the following information. Not all required headers are listed. Refer to the Protocol specification IEC 62766-4-1 for a complete list. <table border="1"> <tr> <td>X-OITF-Request-Line</td><td>Identify the HNI-IGI method with the well known PSI (Public Service Identifier) of the scheduled content, e.g. INVITE sip:IPTV_SC_Service@iptv.ericsson.com SIP/2.0</td></tr> <tr> <td>X-OITF-From</td><td>Local defined OITF CurrentUser property, e.g. <sip:family@ims.live.ericsson.com>; tag=1211455936632545012</td></tr> <tr> <td>X-OITF-To</td><td>PSI of the scheduled content, e.g. sip:IPTV_SC_Service@iptv.ericsson.com</td></tr> </table>	X-OITF-Request-Line	Identify the HNI-IGI method with the well known PSI (Public Service Identifier) of the scheduled content, e.g. INVITE sip:IPTV_SC_Service@iptv.ericsson.com SIP/2.0	X-OITF-From	Local defined OITF CurrentUser property, e.g. <sip:family@ims.live.ericsson.com>; tag=1211455936632545012	X-OITF-To	PSI of the scheduled content, e.g. sip:IPTV_SC_Service@iptv.ericsson.com
X-OITF-Request-Line	Identify the HNI-IGI method with the well known PSI (Public Service Identifier) of the scheduled content, e.g. INVITE sip:IPTV_SC_Service@iptv.ericsson.com SIP/2.0						
X-OITF-From	Local defined OITF CurrentUser property, e.g. <sip:family@ims.live.ericsson.com>; tag=1211455936632545012						
X-OITF-To	PSI of the scheduled content, e.g. sip:IPTV_SC_Service@iptv.ericsson.com						

IECNORM.COM : Click to view the full PDF of IEC 62766-5-1:2017

8.2.4.5 Network timeshift of broadcast service

Methods	Procedures						
pause ()	The method has different behaviour if the pause() method has previously been invoked. While the first pause() request sets up the session over HNI-IGI, the subsequent pause() requests simply issue an RTSP PAUSE request. First pause() request The OITF shall generate a session modification request over the HNI-IGI including the modified SDP offer. The SDP offer included by the OITF shall have attributes as relevant to the unicast stream to be setup. The OITF shall provide the following information as part of the scheduled session modification request as described in 7.3.3.1 of IEC 62766-4-1:2017. Not all required headers are listed. Refer to the Protocol specification IEC 62766-4-1 for a complete list. <table border="1"> <tr> <td>X-OITF-Request-Line</td><td>Identify the HNI-IGI method with the well-known PSI (Public Service Identifier) of the scheduled content, e.g. INVITE sip:IptvBroadcast@iptv.ericsson.com SIP/2.0</td></tr> <tr> <td>X-OITF-From</td><td>Local defined OITF CurrentUser property, e.g. <sip:family@ims.live.ericsson.com>; tag=1211455936632545012</td></tr> <tr> <td>X-OITF-To</td><td>PSI of the scheduled content, e.g. sip:IptvBroadcast@iptv.ericsson.com</td></tr> </table> On receiving a successful response to the INVITE request and if the channel has an idType of ID_IPTV_URI, the OITF shall send an IGMP Leave and request on the UNIS-13 as described in 9.2.1.1 of IEC 62766-4-1:2017. Subsequent pause() requests This request causes the OITF to send an RTSP PAUSE message (refer to 8.2.1.2 of IEC 62766-4-1:2017). The RTSP PAUSE message shall include: - the RTSP URL shall be set to the value retrieved from the fmtp:iptv_rtsp h-uri attribute of the SDP answer; - session header shall be set as specified in the SDP answer fmtp:iptv_rtsp h-session attribute. After a successful response to the RTSP PAUSE message has been received, the OITF shall generate a PlaySpeedChanged event indicating a playback speed of 0.	X-OITF-Request-Line	Identify the HNI-IGI method with the well-known PSI (Public Service Identifier) of the scheduled content, e.g. INVITE sip:IptvBroadcast@iptv.ericsson.com SIP/2.0	X-OITF-From	Local defined OITF CurrentUser property, e.g. <sip:family@ims.live.ericsson.com>; tag=1211455936632545012	X-OITF-To	PSI of the scheduled content, e.g. sip:IptvBroadcast@iptv.ericsson.com
X-OITF-Request-Line	Identify the HNI-IGI method with the well-known PSI (Public Service Identifier) of the scheduled content, e.g. INVITE sip:IptvBroadcast@iptv.ericsson.com SIP/2.0						
X-OITF-From	Local defined OITF CurrentUser property, e.g. <sip:family@ims.live.ericsson.com>; tag=1211455936632545012						
X-OITF-To	PSI of the scheduled content, e.g. sip:IptvBroadcast@iptv.ericsson.com						
resume ()	The OITF shall send an RTSP PLAY over UNIS-11 using attribute values received in the SDP from the session modification procedure. The RTSP PLAY is as described in 8.2.1.2 of IEC 62766-4-1:2017. The RTSP fields in the RTSP PLAY message shall be filled as follows: - The RTSP URL shall be set from the SDP h-uri attribute in the case of an absolute URI. The data property shall be updated with the SDP h-uri attribute. If the value of h-uri is a relative URI that is in the form of a media path, then the RTSP absolute URL is constructed by the OITF using the SDP IPAddress (from c-line) and port (from m-line) as the base followed by h-uri value for the media path (e.g. rtsp://10.5.1.72:22554/TV3/823527). - The RTSP URL shall be set from the SDP h-uri attribute in the case of an absolute URI. The data property shall be updated with the SDP h-uri attribute. If the value of h-uri is a relative URI that is in the form of a media path, then the RTSP absolute URL is constructed by the OITF using the SDP IPAddress (from c-line) and port (from m-line) as the base followed by h-uri value for the media path (e.g. rtsp://10.5.1.72:22554/TV3/823527). After a successful response to the RTSP PLAY message has been received, the OITF shall generate a PlaySpeedChanged event indicating the actual playback speed.						

Methods	Procedures						
setSpeed(Number speed)	Sets current speed by using the "Scale" header in the RTSP PLAY as described in 8.2.1.2 of IEC 62766-4-1:2017. After a successful response to the RTSP PLAY message has been received, the OITF shall generate a PlaySpeedChanged event indicating a new playback speed.						
seek(Integer offset, Integer reference)	Sets current play position based on the specified offset from the given reference point, by using the "Range" parameter in the RTSP PLAY as described in 8.2.1.2 of IEC 62766-4-1:2017. After a successful response to the RTSP PLAY message has been received, the OITF shall generate a PlayPositionChanged event indicating the new playback position.						
stopTimeShift()	The OITF shall generate a session modification request over the HNI-IGI including the modified SDP offer. The SDP offer included by the OITF shall have attributes as relevant to the channel as described in Clause C.2 of IEC 62766-4-1:2017. The OITF shall provide the following information as part of the scheduled session modification request as described in 7.3.3.1 of IEC 62766-4-1:2017. Not all required headers are listed. Refer to the Protocol specification IEC 62766-4-1 for a complete list. <table border="1"> <tr> <td>X-OITF-Request-Line</td><td>Identify the HNI-IGI method with the well-known PSI (Public Service Identifier) of the scheduled content, e.g. INVITE sip:IptvBroadcast@iptv.ericsson.com SIP/2.0</td></tr> <tr> <td>X-OITF-From</td><td>Local defined OITF CurrentUser property, e.g. <sip:family@ims.live.ericsson.com>; tag=1211455936632545012</td></tr> <tr> <td>X-OITF-To</td><td>PSI of the scheduled content, e.g. sip:IptvBroadcast@iptv.ericsson.com</td></tr> </table> On receiving a successful response to the INVITE request and if the channel has an idType of ID_IPTV_URI, the OITF shall send an IGMP Join and request on the UNIS-13 as described in 8.1.1.1 of IEC 62766-4-1:2017.	X-OITF-Request-Line	Identify the HNI-IGI method with the well-known PSI (Public Service Identifier) of the scheduled content, e.g. INVITE sip:IptvBroadcast@iptv.ericsson.com SIP/2.0	X-OITF-From	Local defined OITF CurrentUser property, e.g. <sip:family@ims.live.ericsson.com>; tag=1211455936632545012	X-OITF-To	PSI of the scheduled content, e.g. sip:IptvBroadcast@iptv.ericsson.com
X-OITF-Request-Line	Identify the HNI-IGI method with the well-known PSI (Public Service Identifier) of the scheduled content, e.g. INVITE sip:IptvBroadcast@iptv.ericsson.com SIP/2.0						
X-OITF-From	Local defined OITF CurrentUser property, e.g. <sip:family@ims.live.ericsson.com>; tag=1211455936632545012						
X-OITF-To	PSI of the scheduled content, e.g. sip:IptvBroadcast@iptv.ericsson.com						
setChannel(Channel channel, Boolean trickplay, String contentAccessDescriptorURL)	The following procedure is only applicable if Network Timeshift of broadcast service is in progress. The OITF shall generate a session modification request over the HNI-IGI including the modified SDP offer. The SDP offer included by the OITF shall have attributes as relevant to the new channel as described in Clause C.2 of IEC 62766-4-1:2017. The OITF shall provide the following information as part of the scheduled session modification request as described in 7.3.3.1 of IEC 62766-4-1:2017. Not all required headers are listed. Refer to the Protocol specification IEC 62766-4-1 for a complete list. <table border="1"> <tr> <td>X-OITF-Request-Line</td><td>Identify the HNI-IGI method with the well-known PSI (Public Service Identifier) of the scheduled content, e.g. INVITE sip:IptvBroadcast@iptv.ericsson.com SIP/2.0</td></tr> <tr> <td>X-OITF-From</td><td>Local defined OITF CurrentUser property, e.g. <sip:family@ims.live.ericsson.com>; tag=1211455936632545012</td></tr> <tr> <td>X-OITF-To</td><td>PSI of the scheduled content, e.g. sip:IptvBroadcast@iptv.ericsson.com</td></tr> </table> On receiving a successful response to the INVITE request and if the channel has an idType of ID_IPTV_URI, the OITF shall send an IGMP Join and request on the UNIS-13 as described in 9.2.1.1 of IEC 62766-4-1:2017.	X-OITF-Request-Line	Identify the HNI-IGI method with the well-known PSI (Public Service Identifier) of the scheduled content, e.g. INVITE sip:IptvBroadcast@iptv.ericsson.com SIP/2.0	X-OITF-From	Local defined OITF CurrentUser property, e.g. <sip:family@ims.live.ericsson.com>; tag=1211455936632545012	X-OITF-To	PSI of the scheduled content, e.g. sip:IptvBroadcast@iptv.ericsson.com
X-OITF-Request-Line	Identify the HNI-IGI method with the well-known PSI (Public Service Identifier) of the scheduled content, e.g. INVITE sip:IptvBroadcast@iptv.ericsson.com SIP/2.0						
X-OITF-From	Local defined OITF CurrentUser property, e.g. <sip:family@ims.live.ericsson.com>; tag=1211455936632545012						
X-OITF-To	PSI of the scheduled content, e.g. sip:IptvBroadcast@iptv.ericsson.com						
String data	This property holds the RTSP URI from the SDP h-uri attribute. Prior to a successful SIP INVITE, the value is undefined.						

Note that all the remaining properties listed in 8.2.3 shall be supported as described.

8.2.5 Communication services with SIP session management

This subclause provides details of mapping of the DAE APIs to the descriptions provided in the Protocol specification IEC 62766-4-1 for APIs between the OITF and the Network over reference point HNI-IGI for Communication Services with SIP session management.

Methods	Procedures
<code>registerUser(String userId, String pin)</code>	Performs registration with the specified user ID as described in 6.4.6.1 of IEC 62766-4-1:2017.
<code>deRegisterUser(String userId)</code>	Performs de-registration with the specified user ID as described in 6.4.6.1 of IEC 62766-4-1:2017.
<code>subscribeNotification</code> (<code>FeatureTagCollection</code> <code>featureTagCollection</code> , <code>Boolean</code> <code>performUserRegistration</code>)	OITF maintains applications that have subscribed to notifications. If applicable it will send a re-registration to the IG. When new messages arrive at the IG it shall notify the OITF (as defined in 6.5.2.3 of IEC 62766-4-1:2017).
<code>unsubscribeNotification()</code>	This is a local call within OITF to notify that the DAE application shall not receive unsolicited notification. The OITF shall use native code to handle new dialogues. Any feature tag values that were added by the DAE application are removed for the indicated <code>userId</code> since no native code is setup to process the new dialogues for the feature tag values.

8.2.6 CoD unicast streaming over RTP and HTTP

8.2.6.1 General

Subclause 8.2.6 provides details of mapping of the DAE APIs to the descriptions provided in the Protocol specification IEC 62766-4-1 for APIs between the OITF and the Network over reference points UNIS-11 and UNIT-17 for CoD unicast streaming over RTP and HTTP.

Method	Procedures
play(Number speed)	The "speed" parameter is a floating point value indicating the requested playback speed. A value of 1 represents normal playback speed, and other values are relative to this. A "speed" value of zero shall not initiate any procedures. RTSP-RTP The RTSP URL signalled by the "data" attribute shall be used to initiate the process defined in 8.2.1.1.1 of IEC 62766-4-1:2017. The "data" attribute shall furthermore be updated with the new URI after redirection requests (moved). The RTSP PLAY request shall include a "Scale" header set to the value of the "speed" parameter passed to the API. The server will play the stream at the specified speed, if supported. If property <code>oItfNORTSPSessionControl</code> is set to <code>true</code> then the RTSP messages DESCRIBE and SETUP are not used. If the <code>play()</code> method is called with a non-zero speed the property <code>oItfRTSPSessionId</code> is copied to the RTSP <code>SessionId</code> header for the RTSP PLAY request. If the <code>oItfRTSPSessionId</code> is undefined the <code>play()</code> method shall fail. After a successful response to the RTSP PLAY message has been received, the OITF shall generate a <code>PlaySpeedChanged</code> event indicating the actual playback speed. HTTP The HTTP URL signalling by the "data" attribute shall be used to initiate the process defined in 6.3.3.2 of IEC 62766-4-1:2017. The "data" attribute shall furthermore be updated with the new URI after redirection requests (moved). The "speed" parameter shall be passed to the OITF media player, which should attempt to play back the content at the requested speed. If the media player successfully begins to play back the content, the OITF shall generate a <code>PlaySpeedChanged</code> event indicating the actual playback speed.
stop()	RTSP-RTP The OITF shall initiate the process defined in 8.2.1.1.2 of IEC 62766-4-1:2017 except if the property <code>oItfNORTSPSessionControl</code> is set to <code>true</code>. HTTP The OITF shall stop playback. The OITF may close the connection to the server and may clear any buffered content.
seek(Integer pos)	RTSP-RTP If the <code>seek()</code> method is called while the player is in the "playing state", it sets current play position to "pos", by using the "Range" parameter in the RTSP PLAY as described in 8.2.1.1 of IEC 62766-4-1:2017. After a successful response to the RTSP PLAY message has been received, the OITF shall generate a <code>playPositionChanged</code> event indicating a new playback position of "pos". If the <code>seek()</code> method is called while the player is in the "paused" state, the value of <code>playPosition</code> is changed to reflect the new play position. This is the new play position that shall be used for the "Range" parameter of the RTSP PLAY message when playback is resumed. HTTP If the <code>seek()</code> method is called while the player is in the "playing state", the OITF shall attempt to playback from the specified position "pos". It may use the RANGE header as described in 6.3.3.2 of IEC 62766-4-1:2017, as necessary. If the media player successfully begins to play back the content from the specified position, the OITF shall generate a <code>PlayPositionChanged</code> event indicating a new playback position of "pos". If the <code>seek()</code> method is called while the player is in the "paused" state, the value of <code>playPosition</code> is changed to reflect the new play position. This is the new play position from which playback shall be resumed.

Method	Procedures
<code>play()</code>	RTSP-RTP This method causes the OITF to send an RTSP PAUSE message (refer to 8.2.1.2 of IEC 62766-4-1:2017). After a successful response to the RTSP PAUSE message has been received, the OITF shall generate a <code>PlaySpeedChanged</code> event indicating a play speed of 0. HTTP The OITF shall pause playback. If the media player successfully pauses playback, the OITF shall generate a play speed event indicating a <code>PlaySpeedChanged</code> of 0.
<code>next()</code>	Not Supported. Note that track information is not supported in the protocol specification and therefore out of scope.
<code>previous()</code>	Not Supported. Note that track information is not supported in the protocol specification and therefore out of scope.

Property	Procedures
<code>read/write String data</code>	RTSP-RTP This property holds the RTSP URI for the content item. HTTP The property holds the HTTP URI for the content item. If the "data" property of the A/V Control object refers to a Content-Access Streaming Descriptor (i.e. the object has type "application/vnd.oipf.ContentAccessStreaming+xml" as defined in 7.14.2.6), the OITF shall perform the following steps prior to performing the procedures defined in IEC 62766-4-1 as described for method <code>play()</code>: - An HTTP GET request shall be made with the Request-URI set to the URL of the Content-Access Streaming Descriptor as denoted by the "data" property of the A/V Control object. - After the server has returned a Content Access Streaming Descriptor (i.e. a document with type "application/vnd.oipf.ContentAccessStreaming+xml"), the OITF shall interpret the contents of the Content-Access Streaming Descriptor and choose a URL defined by one of the <ContentURL> elements. The criteria for choosing a URL can be the DRM system supported by the OITF. The URL shall then be used for setting up a Streaming CoD session, after which playback can be started (when the <code>play()</code> method is invoked). The "data" property of the AV object shall be changed to represent the chosen URL. - Based on the information retrieved from the Content-Access Streaming Descriptor, the OITF shall passing the <DRMControlInformation> to the appropriate DRM agent, and should initialize the AV playback, i.e. by loading the correct codecs as identified by the Content-access Streaming Descriptor.
<code>readonly Number playPosition</code>	The property holds the current play position in milliseconds of the media referenced by the <code>data</code> property. For RTSP-RTP, The property value shall be based on the value retrieved using the RTSP GET PARAMETERS method and parameter "position" (refer to 8.2.1.5 of IEC 62766-4-1:2017) adjusted for played duration and used scale. If information is not available the value shall be undefined. Note this may happen at the beginning of playing a video and GET_PARAMETER has not returned a value.

Property	Procedures
readonly Number playspeeds []	For RTSP-RTP, the property holds the available speeds, or referred in RTSP as Scale, to be used to change the playback speed. The property value shall be based on the value retrieved using RTSP GET PARAMETERS method and parameter "scales" (refer to 8.2.1.5 of IEC 62766-4-1:2017). For HTTP, the possible playback speeds are determined by the OITF internal capabilities and buffering model, and the speed at which content is delivered. The OITF may make this information available via this property. If information is not available the value shall be undefined. Note this may happen at the beginning of playing a video and GET_PARAMETER has not returned a value.
readonly Number playtime	The property holds the total duration in milliseconds of the media referenced by the data property. For RTSP-RTP, the property value shall be based on the value retrieved using RTSP GET_PARAMETER method and parameter "duration" (refer to 8.2.1.5 of IEC 62766-4-1:2017). For HTTP, if the data property references a content-access streaming descriptor that includes the optional "Duration" attribute, then the property value shall be derived from the value encoded in that attribute. Otherwise, if the data property references an MPEG DASH MPD and the @mediaPresentationDuration attribute is present, then the property value shall be derived from the value encoded in that attribute. Otherwise, if the data property references a file in the MP4 file format (as defined in 5.2 of IEC 62766-2-1:2016), then - if that file is fragmented, the property value shall be derived from the value indicated in the fragment_duration of the 'mehd' box if that box is present; - if that file is not fragmented, the property value shall be derived from the value indicated in the duration of the 'mvhd' box; otherwise the property value may be determined using the "Content-Length" HTTP header, although it is noted that this method does not work for variable bit rate content. If information is not available the value shall be undefined. Note this may happen at the beginning of playing a video and GET_PARAMETER has not returned a value.
readonly Number playstate	No procedures defined since it is not related to protocol specification.
readonly Number error	No procedures defined since it is not related to protocol specification.
readonly Number speed	Float value indicating the actual playback speed of the player for the content referenced by the data property. The normal default playback speed is represented by value 1.

Intrinsic event	Procedure
onPlaySpeedChanged	For RTSP-RTP, when RTSP ANNOUNCE with either beginning-of-stream or end-of-stream codes arrives the OITF shall generate onPlaySpeedChanged event with a speed value of 0.
onPlayPositionChanged	For RTSP-RTP, when the response to the RTSP PLAY with Range header request (Range is included when performing seek() with a position) the OITF shall generate onPlayPositionChanged event with the accepted position.

8.2.6.2 CoD media queuing

This subclause extends the mapping defined in 8.2.6.1 to address behaviour when media queuing is in effect.

Methods	Procedures
queue (String uri)	Queued media items available via HTTP or stored on the terminal may be pre-buffered by the OITF in order to reduce transition delays. When pre-buffering media items, the specified buffering policy shall not be affected. For queued media items available via RTSP, session setup may be carried out prior to the end of the currently playing media item.
play (Number speed)	When the start of a media item is reached due to a negative play speed, the playback should resume at normal play speed without playing any previous media items. When the end of a media item is reached, playback of any queued media items shall be initiated automatically at the specified play speed. The OITF shall map this on to the underlying protocol (HTTP or RTSP) as the following sequence of DAE method calls: data = <URI of the queued media item>; play (<current play speed>);
seek (Integer pos)	If the value of pos is outside the current media item, the play position shall not be changed.
read/write String data	Modification of this property shall cause any queued media items to be discarded.

8.2.7 Scheduled content multicast streaming

8.2.7.1 General

Subclause 8.2.7 provides details of mapping of the DAE APIs to the descriptions provided in the Protocol specification IEC 62766-4-1 for APIs between the OITF and the Network over reference points UNIS-11, UNIS-13 and UNIS-17 for Scheduled Content multicast streaming.

8.2.7.2 Conveyance of channel list

Service discovery description procedure as described in 7.2.4.1 of IEC 62766-4-1:2017 enables the OITF to obtain the URL to access the broadcast channel information. The OITF shall use UNIS-7 using this URL to obtain the Broadcast Discovery Record.

8.2.7.3 Switching channels

Methods	Procedures
setChannel (Channel channel, Boolean trickplay, String contentAccessDescriptorURL)	The setChannel method of the video/broadcast object shall be used to initiate a broadcast session or switch channels. If the channel has an idType of ID_IPTV_URI , the OITF shall send an IGMP Leave and an IGMP Join request on the UNIS-13 as described in 9.2.1.1 of IEC 62766-4-1:2017.

8.2.7.4 End broadcast service

Methods	Procedures
release ()	The release method of the video/broadcast object causes the OITF to perform an IGMP Leave on the active broadcast session as described in 9.2.1.1 of IEC 62766-4-1:2017.

8.2.7.5 Network timeshift of broadcast services

Methods	Procedures
pause()	The pause method of the video/broadcast object causes the OITF to perform an IGMP Leave on the active broadcast session as described in 9.2.1.1 of IEC 62766-4-1:2017.
resume()	The RTSP URL signalled by the data attribute shall be used to initiate the process defined in 8.2.1.1.1 of IEC 62766-4-1:2017. The "data" attribute shall furthermore be updated with the new URI after redirection requests (moved). The value of the "scale" header in the RTSP PLAY message shall be the value set by the most recent call to setSpeed(), or 1.0 if the most recent call to setSpeed() set the playback speed to 0 or setSpeed() has not been called. If property oitfNoRTSPSessionControl is set to true then the RTSP messages DESCRIBE and SETUP are not used. If the play() method is called with a non-zero speed the property oitfRTSPSessionId is copied to the RTSP SessionId header for the RTSP PLAY request. If the oitfRTSPSessionId is undefined the play() method shall fail. After a successful response to the RTSP PLAY message has been received, the OITF shall generate a PlaySpeedChanged event indicating the actual playback speed.
setSpeed(Number speed)	Sets current speed by using the "Scale" header in the RTSP PLAY as described in 9.2.1.1 of IEC 62766-4-1:2017. After a successful response to the RTSP PLAY message has been received, the OITF shall generate a PlaySpeedChanged event indicating a new playback speed. If playback is previously paused (either by a call to pause() or by setting the playback speed to 0) then the new speed shall not be applied until the resume() method is called, as described above.
seek(Integer offset, Integer reference)	Sets current play position based on the specified offset from the given reference point, by using the "Range" parameter in the RTSP PLAY as described in 8.2.1.1 of IEC 62766-4-1:2017. After a successful response to the RTSP PLAY message has been received, the OITF shall generate a PlayPositionChanged event indicating the new playback position.
stopTimeshift()	The setChannel() method of the video/broadcast object shall be used to initiate a broadcast session. If the channel has an idType of ID_IPTV_URI , the OITF shall send an IGMP Join request on the UNIS-13 as described in 9.2.1.1 of IEC 62766-4-1:2017.

read/write String data	This property holds the RTSP URI for the content item.

All the remaining properties listed under 8.2.6 shall be supported as described.

8.3 URI schemes and their usage

8.3.1 General

The following table lists possible URL schemas and their usages within DAE documents (XHTML, JavaScript, images, and references to A/V content). If a certain URL scheme is supported, the corresponding protocols to a URL scheme shall be supported as defined by the reference(s).

Table 14 – URI schemes and usages

URI scheme	Usage	Reference	Comments
dvb-mcast	Scheduled content delivery	DVB-MCAST URI scheme as defined by Clause A.1 of ETSI TS 102 539	A URL to refer to a scheduled content channel supported by the OITF and delivered via multicast.
Dvb	Application launching	Locator for applications in SD&S as defined by 6.3.3 of ETSI TS 102 851	The orgid and appid encoded in the DVB URI are compared with the applications signalled in SD&S to identify one with the same orgid and appid.
	Non-realtime downloaded content	Clause 6 of ETSI TS 102 851	Non-realtime downloaded content
Igmp	Scheduled content	Annex F of IEC 62766-4-1	The transport IP Multicast Address to access the service as defined in ETSI TS 102 034
http and https	Transport of DAE documents	Subclause 5.3.3.1 of IEC 62766-4-1 Subclause 5.3 of CEA-2014-A Clause 5 of IEC 62766-7:2017	A URL to refer documents supported by DAE.
	Unicast streaming or download over HTTP	HAS specification IEC 62766-2-2	A URL to refer to the MPD.
		Annex D of IEC 62766-4-1:2017	A Content URL specified in the data attribute of A/V Control object as defined in 7.14.
crid	COD streaming	Subclause 5.4 of IEC 62766-3:2016	A Content URL specified in a Content Access Descriptor described in Annex E.
	Programme identification via BCG	Subclause 5.4 of IEC 62766-3:2016	
sip	Unicast streaming over RTP ("sip-rtsp-rtp-udp")	Annex D of IEC 62766-4-1:2017	
	Unicast streaming over UDP ("sip-rtsp-udp")		
rtsp	Unicast streaming control		
URI scheme for recordings	Recordings	Subclause 7.10.6.2.	The format of the URI is outside the scope of this document except that: - The scheme shall not be one that is included in this document - The URI shall not include a fragment
URI scheme for downloaded content	Downloaded content	Subclause 7.4.5.2.	The format of the URI is outside the scope of this document except that: - The scheme shall not be one that is included in this document - The not shall not include a fragment

8.3.2 Media fragments support

Clause 8 of IEC 62766-5-2:2017 requires support for temporal clipping based on Normal Play Time as defined in 4.2.1 of the Media Fragments URI specification. URLs including these temporal clipping fragments shall be supported as follows;

- HTTP URIs including a temporal clipping fragment shall be supported. Mapping from time to byte position as defined in 2.1 of W3C Protocol for Media Fragments 1.0 should be supported. Mapping of normal play time to byte ranges in the server defined in 2.2 of W3C Protocol for Media Fragments 1.0 may be supported.
- If the OITF supports RTSP, then RTSP URIs including a temporal clipping fragment shall be supported. The normal play time specified in the URI shall be used by the OITF to determine the starting time and ending time of the media. The normal play time should be mapped to the RTSP protocol as defined in Annex A of W3C Protocol for Media Fragments 1.0.
- If the OITF has local PVR capability then the OITF shall support use of temporal clipping fragments with the URI returned by the 'uri' property of the Recording class. This document is intentionally silent about how mapping from normal play time to byte positions within the recorded content is performed.
- If the OITF has support for downloading content then the OITF shall support use of temporal clipping fragments with the URI returned by the 'uri' property of the Download class. This document is intentionally silent about how mapping from normal play time to byte positions within the downloaded content is performed.

The begin time shall behave as start-of-media and the end time shall behave as end-of-media. If the value of temporal fragment interval is changed, then there will be no change in the play state unless the interval is changed so that the current play position is outside the interval.

8.4 Mapping from APIs to content formats

8.4.1 Character conversion

Except for the `getSIDescriptors()` method (see 7.16.2.5), the OITF shall translate all characters extracted from DVB SI tables and descriptors into their UTF-16 equivalent when exposing the character in a JavaScript character or string object. In addition, the following rules shall apply:

- the character table of text fields in DVB SI shall be determined as specified in Annex A of ETSI EN 300 468; the default character table may be determined by the local broadcast system;
- the bytes denoting the character table and the control codes for character emphasis on and off shall be filtered out by the OITF;
- the control codes for "CR/LF" shall be expanded to the two separate UTF-16 characters U+000D and U+000A.

8.4.2 AVComponent

AVComponent objects represent the components in a stream. For an MPEG-2 transport stream not delivered via adaptive streaming, the set of components shall be the audio, video and subtitle components listed in the PMT of the service. For content in the MP4 file format not delivered via adaptive streaming, the set of components shall be the audio, video and subtitle tracks listed in the "moov" box. For content delivered via adaptive streaming, A/V Component objects shall correspond to adaptation sets in the MPD.

The following table shows the mapping from the properties of the AVComponent class to the data carried inside the MPEG-2 TS and MP4 file format.

Property name and type	MPEG-2 TS with DVB-SI component descriptor in SDT and/or EIT	MPEG-2 TS without DVB-SI SDT and EIT	MP4 FF	MPEG DASH
Name: componentTag Type: Integer	The contents of the component_tag field in the stream_identifier_descriptor in PMT		Not defined	The value of the id attribute in the AdaptationSet (if provided)
Name: pid Type: Integer	The PID of the stream in the PMT		trackID	Not defined
Name: Type Type: One of the following constants COMPONENT_TYPE_VIDEO / COMPONENT_TYPE_AUDIO / COMPONENT_TYPE_SUBTITLE	May be derived as follows: • A value of 0x02 or 0x1B in the stream_type field in the PMT → VIDEO. • A value of 0x03 or 0x11 in the stream_type field in the PMT → AUDIO. • A value of 0x06 in the stream_type field in the PMT and the presence of a DTS_audio_stream_descriptor in the ES loop in the PMT → AUDIO. • A value of 0x06 in the stream_type field in the PMT and the presence of an AC3_descriptor or an Enhanced_AC3_descriptor in the ES loop in the PMT → AUDIO. • A value of 0x06 in the stream_type field in the PMT and the presence of a subtitling_descriptor in the ES loop in the PMT → SUBTITLES. • A value of 0x06 in the stream_type field in the PMT and the presence of a teletext_descriptor in the ES loop in the PMT and an entry in that descriptor with Teletext_type set to 0x02 or 0x05 → SUBTITLES.		Track has a VisualSampleEntry (handler_type = "vide") -> COMPONENT_TYPE_VIDEO Track has an AudioSampleEntry (handler_type = "soun") -> COMPONENT_TYPE_AUDIO	Defined by the value of the '@contentType' attribute
Name: encoding Type: A string identifying the video or audio format as defined in Clause 4 of IEC 62766-2-1:2016 or the subtitle format as defined in 7.16.5.2.1	Property type is COMPONENT_TYPE_VIDEO			Defined by the '@codecs' attribute
	→ "video/mpeg" or "video/mp2t".		Track has a sample description type "avc1" → "video/mp4".	
	Property type is COMPONENT_TYPE_AUDIO			
	A value of 0x03 in the stream_type field in the PMT → "audio/mpeg". A value of 0x11 in the stream_type field in the PMT and the profile_and_level field in that descriptor indicates MPEG-4 AAC or MPEG-4 HE AAC → "audio/mp4". A value of 0x11 in the stream_type field in the PMT and the profile_and_level field in that descriptor indicates MPEG-4 HE AAC v2 → "audio/aacp". A value of 0x06 in the stream_type field in the PMT and the presence of a DTS_audio_stream_descriptor in the ES loop in the PMT → "audio/vnd.dts". A value of 0x06 in the stream_type field in the PMT and the presence of an AC3_descriptor in the ES loop in the PMT → "audio/ac3".		Track has a sample description type "mp4a" → "audio/mp4"	

Property name and type	MPEG-2 TS with DVB-SI component descriptor in SDT and/or EIT	MPEG-2 TS without DVB-SI SDT and EIT	MP4 FF	MPEG DASH
	Property type is COMPONENT_TYPE_SUBTITLE			
	A value of 0x01 in subtitling_type field of the subtitling_descriptor in the ES loop of the PMT → "EBU-SUBT". A value of 0x10 or 0x11 or 0x12 or 0x14 in the subtitling_type field of the subtitling_descriptor in the ES loop of the PMT → "DVB-SUBT". A value of 0x20 or 0x21 or 0x22 or 0x24 in the subtitling_type field of the subtitling_descriptor in the ES loop of the PMT → "DVB_SUBT" (the hearingImpaired property in the derived AVSubtitleComponent would be set to true). The PMT contains a caption_service_descriptor with a digital_cc flag having the value of 1 for at least one of the represented caption services → "CEA-SUBT".		Track has a handler-type "text" → "3GPP-TT".	
Name: encrypted Type: Boolean	May be derived from any of the following: • Presence of a CA_descriptor in the PMT in the program information loop. • Presence of a CA_descriptor in the PMT in the elementary stream information loop describing the stream.		Not defined	
Name: aspectRatio Type: Number containing width divided by height as a decimal Only defined for video components.	Derived from the stream_content and component_type fields in the component_descriptor.	Undefined	Not defined	Defined by the value of the '@par' attribute
Name: language Type: String containing an ISO 639-2 language code as defined in ISO 639-2 Only defined for audio and subtitle components.	Property type is COMPONENT_TYPE_AUDIO			Defined by the value of the '@lang' attribute in the MPD, whether set explicitly or inherited. The contents of the language field in the "mdhd" of the track shall be ignored.
	For audio components, the contents of the ISO_639_language_code field in the ISO_639_language_descriptor in the ES loop of the PMT unless overridden by the ISO_639_language_code field in the supplementary_audio_descriptor.		The contents of the language field in the media header "mdhd" of the track.	
	Property type is COMPONENT_TYPE_SUBTITLE			
	For subtitles, the contents of the ISO_639_language_code field in the subtitling_descriptor or teletext_descriptor, as appropriate.		The contents of the language field in the media header "mdhd" of the track.	

Property name and type	MPEG-2 TS with DVB-SI component descriptor in SDT and/or EIT	MPEG-2 TS without DVB-SI SDT and EIT	MP4 FF	MPEG DASH
Name: audioDescription Type: Boolean – True if component is an audio description Only defined for audio components.	True if any of the following is true: - there is an audio component with an ISO_639_language_descriptor in the PMT with the audio_type field set to 0x03; - there is a supplementary_audio_descriptor with the editorial_classification field set to 0x01; - there is an ac-3_descriptor or an enhanced_ac-3_descriptor with a component_type field with the service_type flags set to Visually Impaired. Otherwise, false.		Not defined	
Name: audioChannels Type: Number indicating 5 for 5.1, 7 for 7.1, 2 for stereo Only defined for audio components.			Not defined	Derived from the contents of the Audio Channel Configuration element
Name: hearingImpaired Type: Boolean – Has value true if the stream is intended for the hearing-impaired (e.g. contains a written description of the sound effects), false otherwise. Only defined for subtitle components.	True if one of the following is true: - There is a subtitling_descriptor with the subtitling_type field set to 0x20, 0x21, 0x22, 0x23 or 0x24. - There is a teletext_descriptor with a teletext_type field with a value of 0x05.		Not defined	Not defined

NOTE This document intentionally does not define a mapping from the properties of the AVComponent class to the HAS MPD.

8.4.3 Channel

Channel objects represent data streams carrying content that the OITF can tune to. In some cases the channel object may have been explicitly created by an application, but usually they will have been created when the OITF discovers the channel when performing a scan or reading an SD&S file. The following tables show the mapping from the properties of the Channel class to the source of the data for that property.

All references in the tables to the SDT are for the SDT Actual table (i.e. the SDT carried in the MPEG2-TS with a PID value of 0x0011 and a table_id value of 0x42, as defined in ETSI EN 300 468), and references to the BroadcastDiscovery and PackageDiscovery are to the elements of those names in SD&S.

For channels of type ID_DVB_*:

Property name	Source	Comment
channelType	Assigned by the terminal.	Assigned by the terminal to TYPE_TV or TYPE_RADIO based on the service type signalled in SDT/service descriptor/service type or undefined otherwise.
idType	Assigned by the terminal or by the application.	Assigned by the terminal based on the type of channel, if the channel was discovered by a channel scan, or by the application using the value passed in the <code>createChannelObject()</code> method.
ccid	Assigned by the terminal.	Unique identifier for the channel
tunerID	Assigned by the terminal.	Unique identifier for the tuner
onid	Assigned by the terminal or by the application.	Assigned by the terminal from SDT.onid or by the application using the value passed in to the <code>createChannelObject()</code> method.
nid	Assigned by the terminal	Assigned by the terminal as follows: If during the terminal configuration process, a network_id value was selected (either explicitly or implicitly) and the NIT subtable with that network_id value was used by the terminal to discover the correct delivery system descriptor of this channel, then the value of this property shall be that network_id value. Otherwise, if there is exactly one NIT 'actual' subtable in the Transport Stream that is carrying the channel, then the value of this property shall be the network_id in that subtable. Terminals are not required to update the value if it changes dynamically in the broadcast Transport Stream. Otherwise the value shall be undefined .
tsid	Assigned by the terminal or by the application.	Assigned by the terminal from SDT.tsid or PAT.tsid or by the application using the value passed in to the <code>createChannelObject()</code> method.
sid	Assigned by the terminal or by the application.	Assigned by the terminal from SDT.sid or by the application using the value passed in to the <code>createChannelObject()</code> method.
sourceID	Assigned by the terminal.	Takes the value undefined
freq	Assigned by the terminal.	Takes the value undefined
cni	Assigned by the terminal.	Takes the value undefined
name	Assigned by the terminal.	Assigned by the terminal from SDT/service descriptor/service name or undefined for Channel objects created by calls to the <code>createChannelObject()</code> method.
majorChannel	Assigned by the terminal.	Either takes the value undefined or, in markets where logical numbers are used, takes the value of the logical channel number for the channel as signalled in the broadcast specification for that market.
minorChannel	Assigned by the terminal.	Takes the value undefined
dsd	Assigned by the terminal or by the application.	Assigned by the application using the delivery system descriptor passed in to the <code>createChannelObject()</code> method, or implementation dependent in all other cases.
favourite	Assigned by the terminal.	
favIDs	Assigned by the terminal.	
locked	Assigned by the terminal.	
manualBlock	Assigned by the terminal.	
ipBroadcastID	Assigned by the terminal.	Takes the value undefined
channelMaxBitRate	Assigned by the terminal.	Takes the value undefined

Property name	Source	Comment
channelTTR	Assigned by the terminal.	Takes the value undefined
recordable	Assigned by the terminal.	Implementation dependent
longName	Assigned by the terminal.	Implementation dependent
description	Assigned by the terminal.	Implementation dependent
authorised	Assigned by the terminal.	Implementation dependent
genre	Assigned by the terminal.	Implementation dependent
hidden	Assigned by the terminal or by the application.	If the DVB broadcast system supports a logical channel number mechanism that can identify channels that are not expected to be offered to the user in a channel list then the value of this property should be derived from that signalling. Otherwise the value of this property is implementation dependent. NOTE This document does not itself include a logical channel number mechanism for channels of type ID_DVB_*.
logoURL	Assigned by the terminal.	Implementation dependent
isHD	Assigned by the terminal	Assigned by the terminal to true or false based on the service type signalled in SDT/service descriptor/service type or undefined otherwise.
is3D	Assigned by the terminal	Assigned by the terminal to true or false based on the service type signalled in SDT/service descriptor/service type or undefined otherwise.

For channels of type ID_IPTV_SDS:

Property name	Source	Comment
channelType	Assigned by the terminal.	Assigned by the OITF based on the value signalled in SDT/service descriptor/service type in the stream if BroadcastDiscovery/ServiceList/SingleService/Sl@PrimarySISource is "Stream", otherwise assigned based on the value of BroadcastDiscovery/ServiceList/SingleService/Sl@ServiceType (if present). Otherwise, or if not known, set to undefined .
idType	Assigned by the terminal or by the application.	Assigned by the OITF to ID_IPTV_SDS if the channel was discovered using SD&S metadata, or assigned by the application using the value passed in the createChannelObject() method.
ccid	Assigned by the terminal.	Unique identifier for the channel
tunerID	Assigned by the terminal.	Unique identifier for the tuner if relevant or set to undefined
onid	Assigned by the terminal.	Assigned by the OITF to the value signalled in BroadcastDiscovery/ServiceList/SingleService/DVBTriplet@OrigNetId
nid	Assigned by the terminal.	Implementation dependent.
tsid	Assigned by the terminal.	Assigned by the OITF to the value signalled in BroadcastDiscovery/ServiceList/SingleService/DVBTriplet@TSId
sid	Assigned by the terminal.	Assigned by the OITF to the value signalled in BroadcastDiscovery/ServiceList/SingleService/DVBTriplet@ServiceId
sourceID	Assigned by the terminal.	Takes the value undefined
freq	Assigned by the terminal.	Takes the value undefined

Property name	Source	Comment
cni	Assigned by the terminal.	Takes the value undefined
name	Assigned by the terminal.	Assigned by the OITF from SDT/service descriptor/service name in the stream if BroadcastDiscovery/ServiceList/SingleService/Sl@PrimaryS ISource is "Stream", otherwise set to BroadcastDiscovery/ServiceList/SingleService/Sl/Name (if present), otherwise set to BroadcastDiscovery/ServiceList/SingleService/TextualIdentifier@ServiceName
majorChannel	Assigned by the terminal.	Assigned by the OITF from PackageDiscovery/Package/Service/LogicalChannelNumber (if present), otherwise takes the value undefined
minorChannel	Assigned by the terminal.	Takes the value undefined
dsd	Assigned by the terminal.	Takes the value undefined
favourite	Assigned by the terminal.	
favIDs	Assigned by the terminal.	
locked	Assigned by the terminal.	
manualBlock	Assigned by the terminal.	
ipBroadcastID	Assigned by the terminal or by the application.	Assigned by the OITF to the DVB textual service identifier of the IP broadcast service, specified in the format "ServiceName.DomainName" with the ServiceName and DomainName taken from the attributes of BroadcastDiscovery/ServiceList/SingleService/TextualIdentifier, or assigned by the application using the value passed in to the createChannelObject() method
channelMaxBitRate	Assigned by the terminal.	Assigned by the OITF to the value provided in BroadcastDiscovery/ServiceList/SingleService/MaxBitrate (if present), otherwise undefined
channelTTR	Assigned by the terminal.	Assigned by the OITF to the value provided in BroadcastDiscovery/ServiceList/SingleService/TimeToRenegotiate (if present), otherwise undefined
recordable	Assigned by the terminal.	Implementation dependent
longName	Assigned by the terminal.	Set by the OITF to the Name element that is a child of the BCG ServiceInformation element describing the channel, where the length attribute of the Name element has the value 'long'
description	Assigned by the terminal.	Set by the OITF to BroadcastDiscovery/ServiceList/SingleService/Sl/Description (if present), otherwise set to the ServiceDescription element that is a child of the BCG ServiceInformation element describing this channel.
authorised	Assigned by the terminal.	Implementation dependent
genre	Assigned by the terminal.	Set by the OITF to BroadcastDiscovery/ServiceList/SingleService/Sl/ContentGenre (if present), otherwise set to the values of any ServiceGenre elements that are children of the BCG ServiceInformation element describing the channel.
hidden	Assigned by the terminal or by the application.	Implementation dependent
logoURL	Assigned by the terminal.	Set by the OITF to the value of the first Logo element that is a child of the BCG ServiceInformation element describing the channel, when this element specifies the URL of an image.

Property name	Source	Comment
isHD	Assigned by the terminal.	Assigned by the terminal to true or false based on the service type signalled in SDT/service descriptor/service type if BroadcastDiscovery/ServiceList/SingleService/Sl@PrimaryS ISource is "Stream", otherwise assigned based on the value of BroadcastDiscovery/ServiceList/SingleService/Sl@ServiceT ype (if present). Otherwise, or if not known, set to undefined .
is3D	Assigned by the terminal.	Assigned by the terminal to true or false based on the service type signalled in SDT/service descriptor/service type if BroadcastDiscovery/ServiceList/SingleService/Sl@PrimaryS ISource is " Stream ", otherwise assigned based on the value of BroadcastDiscovery/ServiceList/SingleService/Sl@ServiceT ype (if present). Otherwise, or if not known, set to undeFined .

For channels of type ID_IPTV_URI:

Property name	Source	Comment
channelType	Assigned by the terminal.	Takes the value undefined .
idType	Assigned by the application.	Assigned by the application using the value passed in the createChannelObject() method.
ccid	Assigned by the terminal.	Unique identifier for the channel
tunerID	Assigned by the terminal.	Unique identifier for the tuner if relevant or set to undefined
onid	Assigned by the terminal or by the application.	Assigned by the application using the value passed in to the createChannelObject() method
nid	Assigned by the terminal	Implementation dependent
tsid	Assigned by the terminal or by the application.	Assigned by the application using the value passed in to the createChannelObject() method
sid	Assigned by the terminal or by the application.	Assigned by the application using the value passed in to the createChannelObject() method
sourceID	Assigned by the terminal.	Takes the value undefined
freq	Assigned by the terminal.	Takes the value undefined
cni	Assigned by the terminal.	Takes the value undefined
name	Assigned by the terminal.	Takes the value undefined
majorChannel	Assigned by the terminal.	Takes the value undefined
minorChannel	Assigned by the terminal.	Takes the value undefined
dsd	Assigned by the terminal.	Takes the value undefined
favourite	Assigned by the terminal.	
favIDs	Assigned by the terminal.	
locked	Assigned by the terminal.	
manualBlock	Assigned by the terminal.	
ipBroadcastID	Assigned by the terminal.	Assigned by the application using the value passed in to the createChannelObject() method

Property name	Source	Comment
channelMaxBitRate	Assigned by the terminal.	Takes the value undefined
channelTTR	Assigned by the terminal.	Takes the value undefined
recordable	Assigned by the terminal.	Implementation dependent
longName	Assigned by the terminal.	Takes the value undefined
description	Assigned by the terminal.	Takes the value undefined
authorised	Assigned by the terminal.	Implementation dependent
genre	Assigned by the terminal.	Takes the value undefined
hidden	Assigned by the terminal or by the application.	Implementation dependent
logoURL	Assigned by the terminal.	Takes the value undefined
isHD	Assigned by the terminal.	If the channel is being received by the OITF, assigned by the terminal to true or false based on: - For MPEG2-TS content with service_descriptor in SDT, the property takes the value as defined for a channel of type ID_DVB_*. - For content delivered using MPEG-DASH, the property takes the value true if the MPD AdaptationSet element height attribute is set to a value greater than or equal to 720, false otherwise. Otherwise, it takes the value undefined.
is3D	Assigned by the terminal.	If the channel is being received by the OITF, assigned by the terminal to true or false based on: - For MPEG2-TS content with service_descriptor in SDT, the property takes the value as defined for the channel of type ID_DVB_*. - For content delivered using MPEG-DASH, the property takes the value true if 3D video is indicated in the MPD AdaptationSet element FramePacking attribute, false otherwise. Otherwise, it takes the value undefined.

8.4.4 Programme, ScheduledRecording, Recording and Download

The following table defines the mapping between the properties of the Programme, ScheduledRecording, Recording and Download classes.

Property name	Source	Programme Class Property	ScheduledRecording Class Property	Recording Class Property	Download Class Property
state	Assigned by the terminal.	N/A	N/A	Assigned and updated by the terminal as recording is carried out.	Assigned by the terminal, 7.4.5.2 of the present document.
id	Assigned by the terminal.	N/A	N/A	Unique internal identifier for recordings.	Unique internal identifier for downloaded content.
startPadding	Assigned by the terminal or the application.	N/A	Default value assigned by the terminal; may be overridden by the application.	Derived by the terminal from the corresponding property on the ScheduledRecording object.	N/A

Property name	Source	Programme Class Property	ScheduledRecording Class Property	Recording Class Property	Download Class Property
endPadding	Assigned by the terminal or the application.	N/A	Default value assigned by the terminal; may be overridden by the application.	Derived by the terminal from the corresponding property on the ScheduledRecording object.	N/A
repeatDays	Set by the application	N/A	The days on which the recording will be repeated as assigned by the application	Derived by the terminal from the corresponding property on the ScheduledRecording object.	N/A
name	Assigned by the terminal.	Assigned by the terminal from EIT/short_event_descriptor/event name	Derived from Programme object when recording is scheduled. For manual recordings, assigned by the terminal (see note).	Derived by the terminal from the corresponding property on the ScheduledRecording object.	Assigned by the terminal from CADD.Title.
description	Assigned by the terminal.	Assigned by the terminal from EIT/short_event_descriptor/description	Derived from Programme object when recording is scheduled	Derived by the terminal from the corresponding property on the ScheduledRecording object.	Assigned by the terminal from CADD.Synopsis if present.
longDescription	Assigned by the terminal.	Assigned by the terminal from EIT/extended_event_descriptor/text	Derived from Programme object when recording is scheduled	Derived by the terminal from the corresponding property on the ScheduledRecording object.	N/A
startTime	Assigned by the terminal or application.	Assigned by the terminal from EIT/event/start_time.	Derived from Programme object when recording is scheduled. Assigned by the application for recordings scheduled using the recordAt() method. For manual recordings initiated via a native UI, assigned by the terminal (see note).	Derived by the terminal from the corresponding property on the ScheduledRecording object.	Assigned by the terminal based on the startTime argument of RegisterDownload().
recordingStartTime	Assigned by the terminal.	N/A	N/A	The actual start time of the recording.	N/A
timeElapsed	Assigned by the terminal.	N/A	N/A	N/A	Assigned by the terminal.
timeRemaining	Assigned by the terminal.	N/A	N/A	N/A	Assigned by the terminal.

Property name	Source	Programme Class Property	ScheduledRecording Class Property	Recording Class Property	Download Class Property
duration	Assigned by the terminal or application.	Assigned by the terminal from EIT/event/duration.	Derived by the terminal from the duration property of the Programme object when the recording is scheduled. Assigned by the application for recordings scheduled using the recordAt() method. For manual recordings initiated via a native UI, assigned by the terminal (see note).	Derived by the terminal from the corresponding property on the ScheduledRecording object.	N/A
recordingDuration	Assigned by the terminal.	N/A	N/A	The actual duration of the recording.	N/A
channel	Assigned by the terminal.	Reference to broadcast channel where content is available. Set to broadcast content location.	Derived by the terminal from the ccid property of the Programme object when the recording is scheduled. Derived by the terminal from the value passed by the application for recordings scheduled using the recordAt() method. For manual recordings initiated via a native UI, assigned by the terminal (see note).	Derived by the terminal from the corresponding property on the ScheduledRecording object.	N/A
channelId	Assigned by the terminal.	Populated from ccid of the channel carrying this programme.	N/A	N/A	N/A

Property name	Source	Programme Class Property	ScheduledRecording Class Property	Recording Class Property	Download Class Property
programmeID	Assigned by the terminal.	If a programme CRID is not provided in the EIT for the programme then this shall be assigned by the terminal from EIT/event_id and it shall be encoded as a DVB URL referencing a DVB-SI event as enabled by 5.2.4 of IEC 62766-3:2016. Otherwise this is outside the scope of the present document.	Derived from Programme object when recording is scheduled	Derived by the terminal from the corresponding property on the ScheduledRecording object.	N/A
programmeIDType	Assigned by the terminal.	Assigned by the terminal.	Derived from Programme object when recording is scheduled	Derived by the terminal from the corresponding property on the ScheduledRecording object.	N/A
parentalRatings	Assigned by the terminal	Populated from EIT/parental_rating_descriptor/rating, where present.	Derived from Programme object when recording is scheduled. For manual recordings initiated via a native UI, assigned by the terminal (see note).	Derived by the terminal from the parentalRating property on the ScheduledRecording object.	Assigned by the terminal from CADD.parentalRating if present.
contentID	Assigned by the terminal.	N/A	N/A	N/A	Assigned by the terminal from CADD.contentID if present.
totalSize	Assigned by the terminal.	N/A	N/A	N/A	Assigned by the terminal from CADD.contentURL@size, then updated to actual size on disk at end of download.
contentURL	Assigned by the terminal.	N/A	N/A	N/A	Assigned by the terminal from CADD.contentURL.

Property name	Source	Programme Class Property	ScheduledRecording Class Property	Recording Class Property	Download Class Property
drmControl	Assigned by the terminal.	N/A	N/A	N/A	Assigned by the terminal from CADD.DRMControlInformation if present.
transferType	Assigned by the terminal.	N/A	N/A	N/A	Assigned by the terminal from CADD.contentURL.transferType
originSite	Assigned by the terminal.	N/A	N/A	N/A	Assigned by the terminal from CADD.originSite
originSiteName	Assigned by the terminal.	N/A	N/A	N/A	Assigned by the terminal from CADD.originSiteName if present.
iconURL	Assigned by the terminal.	N/A	N/A	N/A	Assigned by the terminal from CADD.iconURL if present.
longName	Assigned by the application	For Programme objects created using the createprogrammeObject() method, this may be set by the application. No standardised mapping in DVB-SI	Derived from Programme object when recording is scheduled.	Derived by the terminal from the longName property on the ScheduledRecording object.	N/A
episode	Assigned by the application	For Programme objects created using the createprogrammeObject() method, this may be set by the application. No standardised mapping in DVB-SI	Derived from Programme object when recording is scheduled.	Derived by the terminal from the episode property on the ScheduledRecording object.	N/A
totalEpisodes	Assigned by the application	For Programme objects created using the createprogrammeObject() method, this may be set by the application. No standardised mapping in DVB-SI	Derived from Programme object when recording is scheduled.	Derived by the terminal from the totalEpisodes property on the ScheduledRecording object.	N/A

Property name	Source	Programme Class Property	ScheduledRecording Class Property	Recording Class Property	Download Class Property
blocked	Assigned by the terminal	Set based on parental control settings for broadcast	N/A	Set based on parental control settings	N/A
showType		No standardised mapping in DVB-SI.	N/A	No standardised mapping in DVB-SI.	N/A
subtitles	Assigned by the terminal	Set in the presence of EIT/subtitle component descriptor for broadcast content for content within schedule.	N/A	Set in the presence of EIT/subtitle component descriptor for broadcast content within scope of schedule when the recording starts.	N/A
isHD	Assigned by the terminal	Set in the presence of an EIT/component descriptor with stream_content value 0x01 or 0x05 and a component_type value indicating "high definition video" as defined in table 26 of ETSI EN 300 468, for broadcast content within scope of schedule.	N/A	Set in the presence of an EIT/component descriptor with stream_content value 0x01 or 0x05 and a component_type value indicating "high definition video" as defined in Table 26 of ETSI EN 300 468, for broadcast content within scope of schedule when the recording starts.	N/A
audioType	Assigned by the terminal	Derived from EIT/component descriptors with stream_content value 0x02, 0x04 or 0x06 for broadcast content within scope of schedule.	N/A	Derived from EIT/component descriptors with stream_content value 0x02, 0x04 or 0x06 for broadcast content within scope of schedule when the recording starts.	N/A
isMultilingual	Assigned by the terminal	Set when the set of language codes for EIT/component descriptors with stream_content value 0x02, 0x04 or 0x06 contains more than one language code for broadcast content within scope of schedule.	N/A	Set when the set of language codes for EIT/component descriptors with stream_content value 0x02, 0x04 or 0x06 contains more than one language code for broadcast content within scope of schedule when the recording starts.	N/A

Property name	Source	Programme Class Property	ScheduledRecording Class Property	Recording Class Property	Download Class Property
genre	Assigned by the terminal	Populated from EIT/content_descriptor/content_nibble_level_1 for broadcast content.	N/A	For broadcast content, populated from EIT/content_descriptor/content_nibble_level_1 when the recording starts.	N/A
hasRecording	Assigned by the terminal	Set if the content item is already recorded on Terminal based storage.	N/A	N/A	N/A
audioLanguages	Assigned by the terminal	Derived from language code(s) present in EIT/component descriptors with stream_content value 0x02, 0x04 or 0x06 for broadcast content within scope of schedule.	N/A	Derived from language code(s) present in EIT/component descriptors with stream_content value 0x02, 0x04 or 0x06 for broadcast content within scope of schedule when the programme is recorded.	N/A
subtitleLanguages	Assigned by the terminal	Derived from language code(s) present in EIT/component descriptors with stream_content value 0x03 for broadcast content within scope of schedule.	N/A	Derived from language code(s) present in EIT/component descriptors with stream_content value 0x03 for broadcast content within scope of schedule when the programme is recorded.	N/A
locked	Assigned by the terminal	Set based on parental control information	N/A	Set based on parental control information	N/A
isManual	Assigned by the terminal	N/A	N/A	Set based on how the recording was scheduled – see the descriptions of the record() and recordAt() methods in 7.10.2.2.	N/A
doNotDelete	Assigned by the application or the terminal	N/A	N/A	May be set by the terminal from a native UI, or by an application.	N/A
saveDays	Assigned by the application or the terminal	N/A	N/A	May be set by the terminal from a native UI, or by an application.	N/A

Property name	Source	Programme Class Property	ScheduledRecording Class Property	Recording Class Property	Download Class Property
saveEpisodes	Assigned by the application or the terminal	N/A	N/A	May be set by the terminal from a native UI, or by an application.	N/A
is3D	Assigned by the terminal	For MPEG-2 TS content with DVB-SI component_descriptor in SDT and/or EIT: True if the component_descriptor in EIT, or if not available, in SDT indicates a 3D video format, including values 0x80, 0x81, 0x82 or 0x83.		For MPEG-2 TS content with DVB-SI component_descriptor in SDT and/or EIT: True if the component_descriptor in EIT, or if not available, in SDT indicates a 3D video format, including values 0x80, 0x81, 0x82 or 0x83.	
isEncrypted	Assigned by the terminal	N/A	N/A	Assigned and updated by the terminal as recording is carried out.	Derived from DRM specific data in content
DRMSystemIds	Assigned by the terminal	N/A	N/A	Assigned and updated by the terminal as recording is carried out.	Derived from DRM specific data in content
getDRMPrivateData	Assigned by the terminal	N/A	N/A	Assigned and updated by the terminal as recording is carried out.	Derived from DRM specific data in content

Where there are multiple language versions of a text field derived from DVB-SI tables, the terminal should select one in accordance with pre-defined user preferences.

The following table shows the mapping from the properties of Download and Recording class to the data carried inside the MPEG-2 TS and MP4 file format.

Property name	MPEG-2 TS With DVB-SI component_descriptor in SDT and/or EIT	MPEG-2 TS Without DVB-SI SDT and EIT	MP4 FF CENC
isEncrypted	True if a CA_descriptor is present in the PMT: - in the program information loop. - or in a elementary stream information loop describing a stream. NOTE Some channels may have a static PMT but the content may not be encrypted. As the transport scrambling control bits in TS packet header are not checked, the property gives only an indication. T		True, if the scheme type set to 'cenc' and scheme version set to 0x00010000 and default_isEncrypted set to 0x1 in the Track Encryption Box (see Note 1)
DRMSystemIds	CA_system_ID in CA_descriptors in the PMT		UUID in SystemID in the Protection System Specific Header box (see Note 2) ^{a)}

Property name	MPEG-2 TS With DVB-SI component_descriptor in SDT and/or EIT	MPEG-2 TS Without DVB-SI SDT and EIT	MP4 FF CENC
getDRMPrivateData	Private data bytes in CA_descriptors in the PMT		Data in the Protection System Specific Header box
a) When the UUID is referring to a DRM managed by the OITF, the DRMSystemId and getDRMPrivateData() mapping may be overridden. For example, Marlin uses the UUID "urn:uid:69F908AF-238 4816-46EA-910C-CD5DCCCB0A3A" in 'cenc' protected files. This UUID is mapped to the OIPF DRMSystemID for Marlin, i.e. "urn:dvb:casystemid:19188".			
NOTE 1 This standard does not handle the case where track common encryption information is overridden at sample level.			
NOTE 2 The DRMSystemId retrieved from an 'cenc' protected file is by default a UID and returned as an UUID URN as defined in IETF RFC 4122. For example "urn:uuid:706D6953-656C-5244-4D48-656164657221".			

8.4.5 Exposing audio description streams as AVComponent objects

Subclause 7.16.5 defines the AVComponent class and the AVAudioComponent class, which defines various properties to describe the audio stream, and 8.4.2 provides information on how these properties are populated. This includes an audioDescription boolean property which is set to true for audio streams that contain an audio commentary for the people with a visual impairment. Audio description (AD) streams which contain such commentary may be delivered to the terminal as either broadcast mix or receiver mix (see ETSI TS 101 154 Annex E for more information on how this is done for MPEG2-TS streams).

Audio streams without audio description and audio streams with broadcast mix audio description shall be exposed to the application using one AVAudioComponent object per audio stream. Broadcast mix audio description streams shall have the audioDescription property set to true.

Receiver mix audio description streams have to be mixed in the terminal with a main audio stream. There may be multiple main audio streams and multiple receiver mix audio descriptions streams. The supported combinations of main audio stream and receiver mix audio description stream shall be determined by the OITF. Each combination shall be exposed to the application as a separate AVAudioComponent object. The properties of this object shall be set as follows:

- audioDescription shall be set to true;
- language shall be set to the language of the audio description stream;
- audioChannels shall be set to the number of audio channels in the combined stream;
- encrypted shall be set to true if either constituent stream is encrypted;
- componentTag and pid shall be set according to the main audio stream;
- type shall be set to COMPONENT_TYPE_AUDIO;
- if the encoding of the constituent streams is the same, then encoding shall be set accordingly otherwise it shall be undefined.

Receiver mix audio description streams shall not be exposed to applications as separate AVAudioComponent objects.

8.4.6 HTML5 media element mapping

The following table defines the mapping that shall be used between the HTML5 AudioTrack and the MPEG-2 transport stream and ISO BMFF system layers.

Attribute	Summary	Systems layer		
		MPEG-2 TS	ISO BMFF	MPEG DASH
readonly attribute DOMString id;	Identifier of the track	The contents of the component_tag field in the stream_identifier_descriptor in PMT.	track_id	The value of the id attribute in the AdaptationSet (if provided)
readonly attribute DOMString kind;	Category of the track	"" unless the audioDescription property of an AVComponent instance for this track would be set to true in which case "description".	""	As defined in the HTML5 specification as referenced by IEC 62766-5-2.
readonly attribute DOMString label;	Label of the track	""	""	""
readonly attribute DOMString language	Language of the track	The primary language subtag in the BCP47 string shall be set as defined in 8.4.2 above for the language property of AVComponent.		
attribute boolean enabled;	Is the track enabled	No mapping is appropriate		

The following table defines the mapping that shall be used between the HTML5 VideoTrack and the MPEG-2 transport stream and ISO BMFF system layers.

Attribute	Summary	Systems layer		
		MPEG-2 TS	ISO BMFF	MPEG DASH
readonly attribute DOMString id;	Identifier of the track	The contents of the component_tag field in the stream_identifier_descriptor in PMT	track_id	The value of the id attribute in the AdaptationSet (if provided)
readonly attribute DOMString kind;	Category of the track	""	""	As defined in the HTML5 specification as referenced by IEC 62766-5-2
readonly attribute DOMString label;	Label of the track	""	""	""
readonly attribute DOMString language	Language of the track	Not defined in this document.		
attribute boolean selected;	Is the track selected	No mapping is appropriate.		

The following table defines the mapping that shall be used between the HTML5 TextTrack and the MPEG-2 transport stream, ISO BMFF and MPEG DASH system layers for streams where the type property of an AVComponent would be COMPONENT_TYPE_SUBTITLE as defined above.

Attribute	Summary	System Layer		
		MPEG-2 TS	ISO BMFF	MPEG DASH
readonly attribute DOMString id;	Identifier of the track	The contents of the component_tag field in the stream_identifier_descriptor in PMT	track_id	The value of the id attribute in the AdaptationSet (if provided)
readonly attribute TextTrackKind kind	Category of the track	"subtitles" unless the hearingImpaired property of an AVComponent instance for this track would be set to true in which case "caption".	"subtitles"	"subtitles"
readonly attribute DOMString label;	Label of the track	""	""	""
readonly attribute DOMString language	Language of the track	The primary language subtag in the BCP47 string shall be set as defined in 8.4.2 above for the language property of AVComponent.		
attribute boolean enabled;	Is the track enabled	No mapping is appropriate		
attribute DOMString inBandMetadataTrackDispatchType	Type of the in-band metadata	"" as defined in HTML5.	"" as defined in HTML5.	"" as defined in HTML5.

In all cases where the values in the above tables cannot be used (because necessary information is not provided or because the value in the table is optional), the HTML5 specification as referenced by IEC 62766-5-2 shall apply.

NOTE This is usually an empty string.

8.5 DLNA RUI remote control function implementation

8.5.1 General

Subclause 8.5 aims to give guidelines to the DAE application developer suggesting how the DAE application should be implemented to use a DLNA remote UI Function, covering the following areas:

- relationship between DAE application and control UI;
- XML UI Listing Provisioning;
- retrieving the Control UI;
- receiving a message (control command) from the remote control device and Responding to a message from the remote control device;
- notification to the remote control device;
- multiple application handling.

Subclauses 8.5.2 to 8.5.7 provide more details and include example code in each case.

8.5.2 Relationship between DAE application and control UI

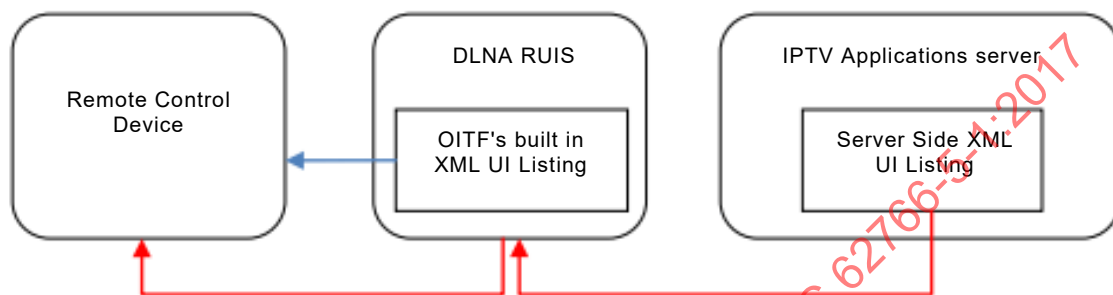
It is assumed that the service provider authors both the DAE application and the control UI to run on the remote control device that communicates with the DAE application. It means that the DAE application and the control UI are managed by one service provider, and the DAE application could handle the HTTP request message which comes from the control UI currently being rendered in the DLNA RUIC.

8.5.3 XML UI listing provisioning

There are two kinds of XML UI Listing (details are described in 5.1.1.5 of CEA-2014-A):

- the OITF's built in XML UI Listing that originates from the OITF (DLNA RUIS) and which is usually pre-defined by the device vendor,
- the Server Side XML UI Listing that is provided by the DAE application and which is defined by the service provider.

Below is a description of where each type of XML UI Listing comes from (see Figure 17).



IEC

Figure 17 – XML UI listing provisioning

- OITF's built in XML UI Listing (blue arrow in Figure 17):
 - this XML UI Listing contains a set of URI pre-defined by the OITF corresponding to a number of Control UIs that are available in the OITF device itself;
 - the OITF shall use this XML UI Listing until a DAE application calls the `useServerSideXMLUIListing()` method.
- Server Side XML UI Listing (red arrows in Figure 17):
 - this XML UI Listing contains both the URIs which identify the control UIs located on the appropriate IPTV Applications server through the pre-defined URI `"/rcf/request_cui"`.

EXAMPLE `/rcf/request_cui?url=www.cui-server.com/avcontrol.html&param1=value1...`

The XML UI Listing is retrieved (or created dynamically) by a DAE application, which then merges the new XML UI Listing with a current XML UI Listing in the DLNA RUIS using the `useServerSideXMLUIListing()` method. The merged XML UI Listing will be located in the DLNA RUIS.

The OITF shall associate all entries in the XML UI Listing added by a DAE application with that application, such that any HTTP requests from a remote control device for the control UI specified by the XML UI Listing entry shall be passed to the corresponding application.

All URIs provided in the XML UI Listing shall start with the pre-defined URI `"/rcf/request_cui"`, which can then be followed by some application-specific parameters. These parameters can be used by the DAE application to identify the Control UI being requested by the remote control device.

The format of the parameters in the URI is out of scope of the DAE specification.

- When the DAE application is terminated, the OITF shall remove any XML UI Listings previously added by the application.

The following example shows the format of the Server Side XML UI Listing. The `<uri>` element in the Server Side XML UI Listing shall start with the value `"/rcf/request_cui"`.

```
<?xml version="1.0" encoding="UTF-8"?>
```

```

<uilist xmlns="urn:schemas-upnp-org:remoteui:uilist-1-0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:schemas-upnp-org:remoteui:uilist-1-0
CompatibleUIs.xsd">
  <ui>
    <uiID>4560-9876-1265-8758</uiID>
    <name>CoD Control UI Type 1</name>
    <description>Controlling the CoD contents</description>
    <protocol shortName="CE-HTML-1.0">
      <uri>/rcf/request_cui?url=http://21.31.24.55:5910/codcui1</uri>
      <protocolInfo>
        <relatedData xmlns="urn:schemas-ce-org:ce-html-server-caps-1-0"
          xsi:schemaLocation="urn:schemas-ce-org:ce-html-server-caps-1-0
ServerProfiles.xsd">
          <profilelist>
            <ui_profile name="MD_UIPROF"/>
          </profilelist>
        </relatedData>
      </protocolInfo>
    </protocol>
  </ui>
  <ui>
    <uiID>2123-3679-3568-2121</uiID>
    <name>CoD Control UI Type 2</name>
    <protocol shortName="CE-HTML-1.0">
      <uri>/rcf/request_cui?url=http://21.31.24.55:5910/codcui2</uri>
      <protocolInfo>
        <relatedData xmlns="urn:schemas-ce-org:ce-html-server-caps-1-0">
          <profilelist>
            <ui_profile name="MD_UIPROF"/>
          </profilelist>
        </relatedData>
      </protocolInfo>
    </protocol>
  </ui>
</uilist>

```

Below is example source code showing how an application can merge a Server Side XML UI Listing that it has retrieved with the OITF's built-in XML UI Listing.

```

var rcMgr;
var xmlhttp;

function init() {
  ...
  rcMgr = document.getElementById("rcfmanager");
  retrieveXMLUIListingFromServer("/iptv_app/xml_location/request_xml?xml=31",
    mergeXMLUIListing);
  ...
}

function retrieveXMLUIListingFromServer(url, callbackFunc) {
  xmlhttp = new XMLHttpRequest();
  xmlhttp.onreadystatechange = function() {
    if (xmlhttp.readyState == 4) {
      if (xmlhttp.status == 200) {
        callbackFunc(xmlhttp.responseText);
      }
    }
  }
  xmlhttp.open("GET", url, true);
  xmlhttp.send(null);
}

function mergeXMLUIListing(xmluiling) {
  rcMgr.useServerSideXMLUIListing(xmluiling, false);
}

<body onload="init();">
<object id="rcfmanager" type="application/oipfRemoteControlFunction"/>
...

```

8.5.4 Retrieving the control UI

The process of retrieving a Control UI based on an OITF's built-in XML UI Listing is described below.

- a) The remote control device sends the request to the DLNA RUIS for the XML UI Listing.
- b) The remote control device presents a UI based on the information in the XML UI Listing. The user selects an entry from the list.
- c) The remote control device sends the HTTP request containing the URI (which has been specified by the OITF itself) to the DLNA RUIS. The OITF returns the Control UI (from its internal memory).
- d) The remote control device presents the Control UI. This Control UI may be an application itself or may be a list of other available applications. In the latter case, the user selects a link from the Control UI.
- e) The remote control device sends the HTTP request containing the URI from the selected link to the DLNA RUIS. The OITF retrieves the DAE application from the IPTV Applications server and executes it.
- f) The DAE application recognises that it needs to get the control UI.
- g) The DAE application retrieves the Control UI from the IPTV Applications server.
- h) The DAE application passes the Control UI received from the IPTV Applications server to the remote control device.

The process of retrieving a Control UI based on a Server Side XML UI Listing is as below:

- a) The remote control device sends the request to the DLNA RUIS for the XML UI Listing.
- b) The remote control device presents a UI based on the information in the XML UI Listing. The user selects an entry from the list.
- c) The remote control device sends the HTTP request containing the URI (which shall start with "/rcf/request_cui") to the OITF DLNA RUIS. The OITF matches the URI with the correct DAE application and passes the request to that DAE application as a `ReceiverRemoteMessage` event.
- d) The DAE application translates the request which came from the Remote control device into a URI.
- e) The DAE application retrieves the Control UI from the IPTV Applications server using this URI.
- f) The DAE application passes the Control UI received from the IPTV Applications server to the Remote control device using `sendRemoteMessage()`.

More details can be found in Annex H.

When the control UI (CE-HTML document) is being rendered in the remote control device, it can retrieve resources (for example, image, css or JavaScript files) directly from the IPTV Applications server over a secure connection. For deployments where the IPTV Applications server is outside the consumer network, the consumer network's WAN gateway shall allow the DLNA RUIC to access the IPTV Applications server to retrieve resources directly. The remote control device that connects to the IPTV Applications server shall implement the Secure Sockets Layer (SSL) Protocol, the Transport Layer Security (TLS) and the "https:" URI scheme, in order to support secure Internet transactions (as defined in 9.1.2).

Below is example source code to show sending the control UI to the remote control device.

```
var rcMgr;
var xmlhttp;
var deviceHandle;
var reqHandles = new Array();

function init() {
    ...
    rcMgr = document.getElementById("rcfmanager");
```

```

    rcMgr.addEventListener("ReceiveRemoteMessage",
    receiveRemoteMessageFromRD, false);

    // check whether the DAE app is launched by the remote control device or
    not
    if (rcMgr.currentRemoteDeviceHandle != undefined) {
        deviceHandle = rcMgr.currentRemoteDeviceHandle;

        retrieveCUIFromServer("/iptv_applications/cui_location/request_cui?
cui=123",
                                sendCUIToRemoteDevice);
    }
    ...
}

function retrieveCUIFromServer(url, callbackFunc){
    xmlhttp = new XMLHttpRequest();
    xmlhttp.onreadystatechange = function() {
        if (xmlhttp.readyState == 4) {
            if(xmlhttp.status == 200){
                callbackFunc(xmlhttp.responseText);
            }
        }
    }
    xmlhttp.open("GET", url, true);
    xmlhttp.setRequestHeader("X-OITF-RCF-User-Agent",
                                rcMgr.getRemoteDeviceUserAgent(deviceHandle));
    xmlhttp.send(null);
}

function sendCUIToRemoteDevice(cuiCEHTML) {
    rcMgr.sendRemoteMessage(remoteDeviceHandle,                reqHandles.shift(),
    cuiCEHTML);
}

function receiveRemoteMessageFromRD(type, remoteDeviceHandle, reqHandle,
requestLine, headers, body) {
    if (type == 0) {
        deviceHandle = remoteDeviceHandle;
        reqHandles.push(reqHandle);

        // retrieve the CUI CE-HTML document from the IPTV Applications
        server
        retrieveCUIFromServer("/iptv_applications/cui_location/request_cui?cui=123",
                                sendCUIToRemoteDevice);
    }
}
<body onload="init();"
<object id="rcfmanager" type="application/oipfRemoteControlFunction"/>
...

```

8.5.5 Receiving and responding to a message between the control UI in the remote control device and OITF

Figure 18 shows the receiving and responding to a message between the control UI in the remote control device and the OITF.

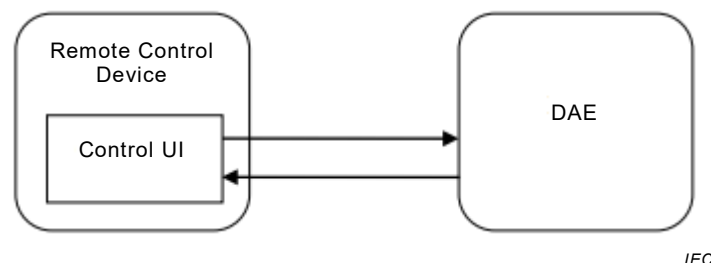


Figure 18 – Remote control message handling

This example shows the usage of receiving and responding to a message between the control UI presented on the remote control device and the OITF. When the control UI sends a message to the DAE application via an HTTP request, the DAE application receives the message via a ReceiveRemoteMessage event. The DAE application shall return the response

to the control UI in the remote control device by using the `sendRemoteMessage()` or `sendInternalServerError()` methods.

The OITF is not able to notify the remote control device whether the DAE application has been terminated or deactivated, or whether the application/`oipfRemoteControlFunction` object has been removed from the application's DOM tree. For this reason, the remote control device may be presenting an outdated copy of the control UI and could send a request from this outdated control UI. In this case, the OITF shall return a 500 response error code to the remote control device.

The OITF shall limit the number of HTTP requests (from the control UI in the remote control device) which have not been responded to by the DAE application. If there are any requests over this limit, the OITF shall automatically reject them and send an HTTP response (HTTP 500 – Internal Server Error) to the remote control device. The OITF shall buffer at least 10 outstanding HTTP requests.

NOTE Clause H.2.4 provides a procedure related to this example.

Below is example source code showing the handling of messages between the DAE application and the control UI that controls the DAE application.

DAE application

```
var rcMgr;
var reqHandles = new Array();

function init() {
    ...
    rcMgr = document.getElementById("rcfmanager");
    rcMgr.addEventListener("ReceiveRemoteMessage", getMessageFromRD, false);
    ...
}

function getMessageFromRD(type, remoteDeviceHandle, reqHandle, requestLine,
headers, body) {
    if (type == 1) {
        // Handling the received message with parameters (requestLine,
headers, body)
        parseAndExecute(body);

        // Sending the proper return message to the remote control device
        var contentType = "Content-Type: Text/plain\n";
        rcMgr.sendRemoteMessage(remoteDeviceHandle, reqHandle, contentType,
"ok");
    }
}

function parseAndExecute(body) {
    // For example, the request from the RD contains the message related to
    // "play of audio" with JSON form (Ex: {'command':415})
    var retVal = eval("(" + body + ")");
    if (retVal.command == VK_PLAY) {
        document.getElementById("aid1").play(1);
    }
}

<body onload="init();">
<object id="rcfmanager" type="application/oipfRemoteControlFunction"/>
<object type="audio/mp4" id="aid1"
data="http://www.avsource.com/audio/bgm.aac">
<param name="loop" value="infinite"/>
</object>
...
```

Control UI

```
var xmlhttp;

function sendPlay() {
    var msg = {'command':415};
    sendMessage("/rcf/request_msg", msg, receiveMsg);
}
```

```

function sendMessage(url, msg, callbackFunc){
    xmlhttp = new XMLHttpRequest();
    xmlhttp.onreadystatechange = function() {
        if (xmlhttp.readyState == 4) {
            if(xmlhttp.status == 200){
                callbackFunc(xmlhttp.responseText);
            }
        }
    }
    xmlhttp.open("POST", url, true);
    request.setRequestHeader("Content-Type", "application/x-www-form-
urlencoded");
    xmlhttp.send(msg);
}

function receiveMsg(msg) {
    alert("Received message from the DAE application: " + msg);
}
<body>
...
<input type="button" value="Play" onclick="javascript:sendPlay();">
...

```

8.5.6 Notification to the remote control device

The application/oipfRemoteControlFunction object supports generating 3rd party multicast notifications and dispatching them to remote control devices. The DAE application can make and send a notification to the remote control devices by using the sendMulticastNotif() method.

If the DAE application wants to send a notification CE-HTML document to all the remote Devices, the DAE application shall set the remoteDeviceHandle parameter in the sendMulticastNotif method to -1.

Otherwise, if the DAE application wants to allow only targeted remote Device (currently being connected to the DAE application) to retrieve the notification CE-HTML document, the DAE application set the proper remoteDeviceHandle parameter in the sendMulticastNotif method when it calls. Then, the OITF shall generate the notification URI with devicehandle and daeid parameters.

If the DAE application wants to send a notification CE-HTML document without storing it in the OITF, the DAE application executes the sendMulticastNotif method with null value in the notifCEHTML parameter. The OITF shall make the notification URI which contains a dynamic parameter with true value, otherwise false is set in the dynamic parameter.

Below is a generated notification URI based on parameter information in the sendMulticastNotif method.

```

?SendToTargetedRD&devicehandle=<target device handle
value>&daeid=<DAE App ID>&dynamic=<true or false>

```

This URL is sent to the remote Devices through the <ruieventURL> element of the multicast notification event and the remote Devices send requests to the OITF with this URL upon receiving it. When the OITF receives the requests from the remote Devices, it shall return the notification CE-HTML document in case the handle of the remote Device which sends the request is the same with the parameter value "<target device handle value>" in the HTTP request URL, otherwise the OITF shall return the HTTP 403 response.

Below is example source code to show that the only targeted remote Device retrieves the notification CE-HTML document.

```

var rcMgr;
var xmlhttp;
var deviceHandle;
var reqHandles = new Array();

function init() {
    ...

```



```

    rcMgr = document.getElementById("rcfmanager");
    rcMgr.addEventListener("ResultMulticastNotif", resultMulticastNotifFromRD,
false);
    ...
}

function sendTargetedNotif() {
    // A remoteDeviceHandle shall be set to -1 if the OITF wants to send the
    // notification CE-HTML UI to all of the remote Devices
    // A remoteDeviceHandle shall be set to a specific value of the device
    handle if
    // the OITF wants to send the notification CE-HTML UI to the targeted
    // remote control devices
    var remoteDeviceHandle = rcMgr.currentRemoteDeviceHandle;
    var eventLevel = 0;
    var notifCEHTML = "<html>...</html>";
    var friendlyName = "Important notification";
    var profilelist = "<ui_profile name='MD_UIPROF'/>";

    rcMgr.sendMulticastNotif(remoteDeviceHandle, eventLevel, notifCEHTML,
friendlyName, profilelist);
}

function resultMulticastNotifFromRD(remoteDeviceHandle, reqHandle, dynamic) {
    if (dynamic != true) {
        alert("Notification is sent to the remote control device well");
    } else {
        //Retrieve a notification CE-HTML UI from server
        ...
    }
}

<body onload="init();">
<object id="rcfmanager" type="application/oipfRemoteControlFunction"/>
...

```

8.5.7 Handling multiple DAE applications and multiple remote control devices

The OITF shall dispatch requests from a remote control device to the DAE application that it is currently controlling. Only one remote control device shall communicate with a DAE application at any time although this could change over time as described below.

- Multiple remote Devices shall not be mapped to a same DAE application at the same time. If a second remote control device attempts to send an HTTP request to a DAE application which is already mapped to a different remote control device, this request shall fail (the OITF sends an HTTP 500 response to the remote control device).
- One remote Device shall not be mapped to multiple DAE applications at the same time. If a remote Device is currently connected to a DAE application and then attempts to make a request to another DAE application, this request shall fail (the OITF sends an HTTP 500 response to the remote Device).

The OITF shall support three mechanisms to drop the connection between a remote control device and a DAE application as follows:

- the remote control device currently bound to the DAE application sends a pre-defined URL `"/rcf/drop_connection"`;
- the DAE application drops the connection with the remote control device by using the `dropConnection()` method;
- the OITF provide a timer mechanism to drop the connection with the remote control device after a period of inactivity (i.e. no HTTP requests received and no HTTP responses sent); the value of the inactivity timer expiry is terminal-specific. One timer will be assigned per remote control device.

If the OITF is unable to dispatch requests to that application (e.g. because the application has terminated or because the `application/oipfRemoteControlFunction` object has been destroyed), the request shall fail (the OITF sends an HTTP 500 response to the remote control device). If the OITF is notified that the remote control device is no longer connected to the network, then the OITF shall allow other remote control devices to connect to the application and act as the new controller of the DAE application.

Figure 19 and Figure 20 give an example showing a mapping relationship between remote control devices and DAE applications.

- remote control device 1 is mapped to DAE application A. The remote control device sends a request to drop the connection with A, using the pre-defined URL "/rcf/drop_connection" and then makes a request to DAE application B. DAE application B responds to the remote control device. The OITF updates its internal state to show that remote control device 1 is now mapped to DAE application B.

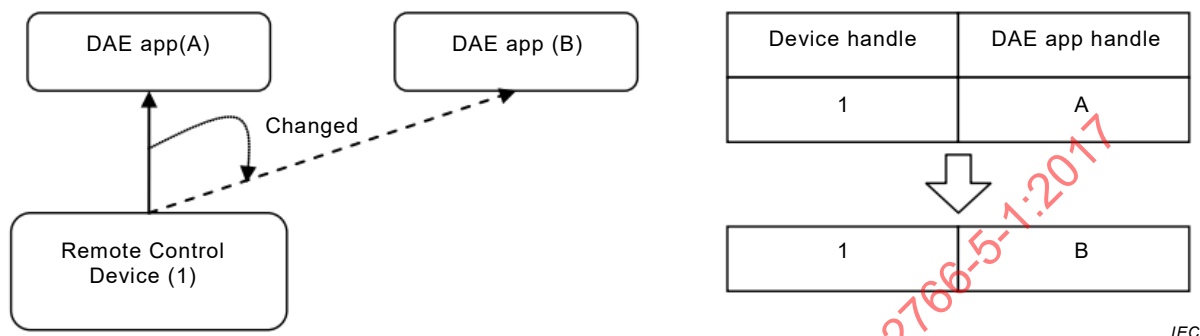


Figure 19 – Remote control device changes mapping between DAE applications

- Remote control device 2 is mapped to DAE application C. A second remote control device 3 then makes a request to DAE application C. The OITF sends an HTTP 500 response to the remote control device 3.

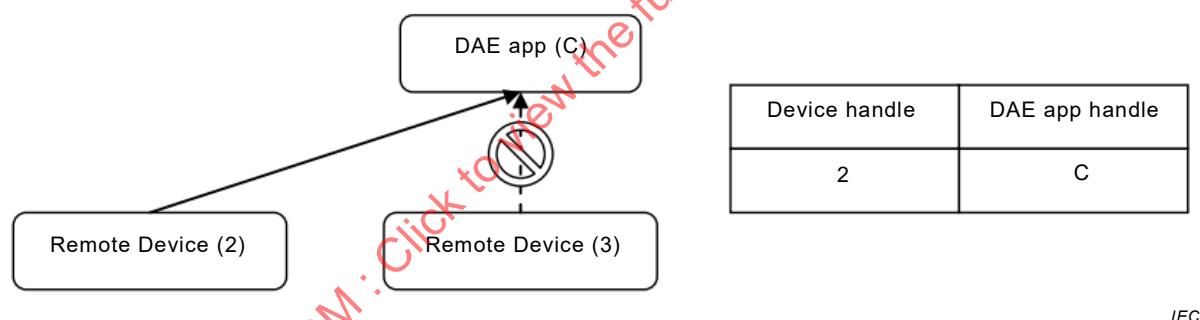


Figure 20 – Remote control device retains control of DAE application

9 Capabilities

9.1 Minimum DAE capability requirements

9.1.1 General

This subclause defines minimum capabilities that OITF implementations are required to provide to the declarative application environment and the applications running in that environment.

OITFs may support multiple simultaneous applications loaded and running in the browser.

When the CEA-2014 notification framework (see 5.3.2) is supported, OITFs shall support at least 2 DAE applications being visible at one time, one application showing a notification in the notification window (as defined in 5.6.3 of CEA-2014-A) and one in the main browser area. OITFs may support more than one DAE application being visible at one time in the main browser area. On OITFs where only one DAE application is visible at one time in the main browser area, it is OITF implementation specific how the visible application is changed.

OITFs with an HD output shall support 1280 × 720 graphics on that output when HD video is being decoded or when no video is being decoded. OITFs may support 1920 × 1080 graphics.

This document does not define any requirements concerning support for SD graphics.

OITFs shall support unrestricted scaling of IP delivered video.

This document does not define any requirements for scaling of video not delivered via IP, e.g. in hybrid OITFs.

This document does not define requirements for supporting decoder format conversion.

This document does not define requirements for pixel depth in the graphics system except that OITFs shall support at least one bit of per-pixel alpha.

This document does not require the capability to mix audio from memory and audio from a currently decoded stream.

OITFs shall support decoding one stream containing video and audio. They may support decoding more than one stream.

The OITF shall support widgets that are of least 100 kB. Widgets of larger size are allowed but the specification remains silent as to the maximum allowed size. When installing a widget with method `installwidget()`, an error message `WIDGET_ERROR_SIZE_EXCEEDED` is returned if the size is exceeded. OITFs shall support the "Tiresias Screenfont" font or equivalent with the "Generic Application Western European Character Set" as defined in Annex C of ETSI TS 102 809. They may support other fonts in addition.

OITFs shall provide some means for text input. This document does not specify any particular solution.

This document recommends support for pointer based input. This document does not define requirements for minimum memory sizes for DAE applications or OITF behaviour when available memory is low. This document is deliberately silent about the conditions under which the LowMemory event defined in subclause 7.2.2.5 is generated.

OITFs shall follow IETF RFC 6265 when implementing cookie support.

Since 6.1 of IETF RFC 6265 does not fix strict limits, this document fixes the following minimum capabilities that terminals shall support:

- at least 4096 bytes per cookie (as measured by the sum of the length of the cookie's name, value, and attributes);
- at least 20 cookies per domain;
- at least 100 cookies total;
- at least 5120 B for the "Set-Cookie" header.

NOTE As implied by IETF RFC 6265, if a cookie or a "Set-Cookie" header is bigger than the maximum size supported by the terminal, it will be discarded, not truncated.

This document does not require control of audio volume to be exposed to the DAE.

The OITF shall include a mechanism for the end user to generate the following key events:

- VK_0 – VK_9
- VK_UP, VK_DOWN, VK_LEFT, VK_RIGHT, VK_ENTER, VK_BACK

- VK_RED, VK_GREEN, VK_YELLOW, VK_BLUE

An OITF shall support the entry of at least a complete set of characters in the range of characters [A-Z], or [a-z] (or both), plus the characters [0-9], plus the characters: space (), full stop (.), comma (,) question mark (?), exclamation mark (!), at (@), colon (:), semi-colon(;), left-bracket ([), right-bracket()], slash(/), minus-sign (-), plus sign (+) and underscore (_), for the textarea element and the input element (taking account of the restrictions defined for the different types of that element).

An OITF may also support a pointer-based interaction paradigm. Terminals that support a free-moving cursor shall indicate this via the "+POINTER" UI Profile Name Fragment as specified in 9.2 and hence shall include `<pointer>true</pointer>` in their XML capabilities.

To provide a good user experience with the widest range of user input devices, DAE applications should make the same feature, function or link accessible via physical keys on the remote also accessible through an element in their user interface, which can be navigated to i) by up, down, left and right (e.g. on a remote control with a very restricted number of buttons) and ii) by a pointer device controlling a free moving cursor on the screen.

If the OITF includes a mechanism to generate the following key events then they shall be available to DAE applications and shall be indicated as part of the capability mechanism defined in Clause 9:

- VK_PLAY, VK_PAUSE, VK_STOP, VK_NEXT, VK_PREV
- VK_PLAY_PAUSE
- VK_FAST_FWD
- VK_REWIND

Some remote controls have separate "play" and "pause" keys; others have a single "play/pause" toggle key. For that reason, in general, it is recommended that applications are written to handle both the VK_PLAY/VK_PAUSE key codes and the VK_PLAY_PAUSE key code.

The OITF may include mechanisms to generate the following key events and if it does, making them available to DAE applications is optional:

- VK_HOME
- VK_MENU
- VK_GUIDE
- VK_TELETEXT
- VK_SUBTITLE
- VK_CHANNEL_UP
- VK_CHANNEL_DOWN
- VK_VOLUME_UP
- VK_VOLUME_DOWN
- VK_MUTE

Where OITFs make other remote control key events available to DAE applications, this shall be done as specified by the capability mechanism defined in Clause 9. Whenever applicable, this should be done using the complementary UI profiles defined in 9.2.