Operator Application Specification

Contents

1 Scope ... 10

2 References ... 10
2.1 Normative references ... 10
2.2 Informative references ... 12

3 Definitions and abbreviations ... 12
3.1 Definitions ... 12
3.2 Abbreviations ... 13

4 Overview ... 14
4.1 Operator applications (informative) ... 14
4.1.1 Scope and motivation ... 14
4.1.2 Types of operator applications ... 15
4.1.3 Standard operator applications ... 15
4.1.3.1 Introduction ... 15
4.1.3.2 Features of standard operator applications ... 15
4.1.3.3 Design policy for standard operator applications ... 15
4.1.3.4 TV channels ... 15
4.1.3.5 Activating and launching of standard operator applications ... 16
4.1.3.6 User input for standard operator applications ... 16
4.1.4 Privileged operator applications ... 16
4.1.5 Operator-specific operator applications ... 17
4.1.6 Coexistence between multiple types of operator application ... 17
4.1.7 Operator applications and channel binding ... 17
4.1.8 Enabling operator applications to be installed and run ... 17
4.1.9 Number of operator applications ... 17
4.2 Architecture (informative) ... 18
4.2.1 Introduction ... 18
4.2.2 System overview ... 18
4.3 Referenced W3C and WHATWG Specifications ... 18

5 User experience (informative) ... 20
5.1 Introduction ... 20
5.2 Using operator applications ... 20
5.2.1 Activation and deactivation ... 20
5.2.2 User input ... 21
5.3 Displaying operator applications ... 22
5.3.1 General visibility of operator applications ... 22
5.3.2 Start page ... 23
5.3.3 Co-existence of operator application and regular HbbTV® applications ... 23
5.3.4 Co-existence of operator application and terminal UI ... 23
5.4 Design policy ... 23
5.4.1 Branding ... 23
5.4.2 User interface design constraints ... 23
5.5 Further concepts ... 24
5.5.1 Channels not offered by the operator ... 24

6 Service and application model ... 24
6.1 Operator application discovery and installation ... 24
6.1.1 Overview ... 24
6.1.2 Triggering operator application discovery ... 26
6.1.3 Operator application discovery methods ... 27
6.1.3.1 Introduction ... 27
6.1.3.2 Broadcast NIT/BAT with URI_linkage_descriptor with operator FQDN ... 27
6.1.3.3 Broadcast NIT/BAT with URI_linkage_descriptor with URI of AIT ... 27
6.1.3.4 NIT from CICAM with uri_linkage_descriptor with URI of XML AIT ... 27
6.1.3.5 Hardwired in terminal with operator FQDN ... 28
6.1.3.6 Hardwired in terminal with URI of XML AIT ... 28
6.1.3.7 DNS SRV lookup to a standardized address ... 28

6.1.4 DNS SRV lookup process 28
6.1.5 (XML) AIT acquisition and download 28
6.1.5.1 XML AIT acquisition 28
6.1.5.2 AIT Acquisition 30
6.1.6 Deciding which operator applications to install 32
6.1.7 Encrypted application package download 32
6.1.7.1 Introduction (informative) 32
6.1.7.2 Encrypted application package download via IP 32
6.1.7.3 Encrypted application package download via DSM-CC object carousel 32
6.1.7.4 Encrypted application package download from CI Plus AuxFS 33
6.1.8 Decrypt, verify, unpack and installation of the application package 33
6.1.9 Installation failures 33
6.1.9.1 Installation failure overview 33
6.1.9.2 Failure handling on first-time installation 34
6.1.9.3 Failure handling when updating an operator application 34
6.2 Updating operator applications 34
6.3 Operator application lifecycle 35
6.3.1 Introduction 35
6.3.2 Starting and stopping operator applications 35
6.3.2.1 Summary (Informative) 35
6.3.2.2 Starting operator applications 36
6.3.2.3 Stopping operator applications 36
6.3.2.4 Co-existence of multiple operator applications 36
6.3.3 Operator application states 37
6.3.3.1 Introduction 37
6.3.3.2 Foreground state 38
6.3.3.3 Background state 39
6.3.3.4 Transient state 40
6.3.3.5 Overlaid foreground state 41
6.3.3.6 Overlaid transient state 41
6.3.4 Standby states 42
6.4 UI elements provided by an operator application 42
6.5 Regular HbbTV® application signalling and lifecycle 43
6.5.1 Application signalling 43
6.5.2 Starting and stopping regular HbbTV® applications 43
6.5.3 Running regular HbbTV® applications with an operator application in the foreground 44
6.5.4 Application signalling for operator application created channels 44
6.6 Multiple operator applications 45
6.6.1 Supported operators 45
6.6.2 Adding operators and operator applications to terminals 45
6.6.3 Installed operator applications 46
6.7 Removal of operator applications 46

7 Formats and protocols 46
7.1 Operator application signalling 46
7.1.1 Launch and startup context signalling 46
7.1.2 Status launch parameter 48
7.2 Extensions to broadcast signalling 49
7.2.1 Application overlay descriptor 49
7.2.2 Application version descriptor 50
7.3 Extensions to broadcast-independent application signalling 50
7.3.1 Minimum application version 50
7.3.2 Operator application images 51
7.4 Operator application ZIP File 52
7.4.1 Operator application ZIP File Format 52
7.4.2 Interoperability Considerations 52
7.4.3 Operator application ZIP File failure conditions 53
7.4.4 Application ZIP file contents 54
7.5 Extensions to DVB-I signalling 54
7.5.1 Operator application controlling media presentation 54
7.6 Metadata for operator application created channels 55
7.6.1 Overview (informative) 55

7.6.2 DVB-I profile 55
8 Browser application environment 55
8.1 Execution model 55
8.2 DAE specification usage 55
8.3 New JavaScript APIs 56
8.3.1 APIs for access to proprietary functions 56
8.3.1.1 The ProprietaryFunctionProvider class 56
8.3.1.2 Deprecated functions 58
8.4 Web APIs 58
8.4.1 Web Notifications 58
8.4.1.1 Requirements 58
8.4.1.2 Usage guidelines 58
8.5 APIs defined in ETSI TS 102 796 59
8.5.1 Modification to terminalChannel 59
9 System integration 59
9.1 Media decoder and tuner resource conflict resolution 59
9.1.1 Overview (informative) 59
9.1.2 Sharing resources for a video/broadcast object 59
9.1.3 Sharing resources for other media decoders 60
9.1.4 Broadcast video presentation and privileged operator applications 60
9.2 Channel lists (informative) 61
9.2.1 Background 61
9.2.2 Operator applications and channel lists 61
9.2.3 CICAM integration 62
9.3 Display model 63
9.4 URLs 63
9.4.1 Origin for an installed operator application 63
9.4.2 Referencing installed operator applications and resources 63
9.5 Access to broadcast carousels 64
9.6 Operator applications and DVB-I 64
9.6.1 Overview (informative) 64
9.6.2 Operator application presentation of DVB-I services 64
9.6.3 Linkage of both regular and operator applications 65
9.6.4 Example usage (informative) 65
9.6.4.1 Example 1 65
9.6.4.2 Example 2 65
9.7 Parental Controls 66
9.7.1 General 66
9.7.2 Examples (Informative) 67
9.8 WebSocket Server and JSON-RPC Implementation 68
9.8.1 WebSocket server 68
9.8.2 Operator application usage 68
9.8.3 API access 68
9.9 Operator application presented channels 69
9.9.1 Overview (informative) 69
9.9.2 Scope 70
9.9.3 Operator application video window 70
9.9.4 API for operator application presented channels 72
9.9.4.1 Introduction (informative) 72
9.9.4.2 JSON-RPC usage 72
9.9.4.2.1 Supported methods 72
9.9.4.2.2 Request capability negotiation 72
9.9.4.2.3 Application and terminal responses 73
9.9.4.3 Concepts 73
9.9.4.3.1 Channel selection and channel sessions 73
9.9.4.3.2 Channel status 73
9.9.4.3.3 Session identifier 74
9.9.4.4 Operator application video window to terminal JSON-RPC requests 74
9.9.4.4.1 org.hbbtv.ipplayback.statusUpdate 74
9.9.4.4.2 org.hbbtv.ipplayback.mediaPositionUpdate 75

9.9.4.4.3 org.hbbtv.ipplayback.setComponents ... 76
9.9.4.4.4 org.hbbtv.ipplayback.setTimelineMapping ... 79
9.9.4.4.5 org.hbbtv.ipplayback.setPresentFollowing ... 80
9.9.4.5 Terminal to operator application video window JSON-RPC requests ... 81
9.9.4.5.1 org.hbbtv.ipplayer.selectChannel ... 81
9.9.4.5.2 org.hbbtv.ipplayer.stop ... 83
9.9.4.5.3 org.hbbtv.ipplayer.play ... 84
9.9.4.5.4 org.hbbtv.ipplayer.setVideoWindow ... 85
9.9.4.5.5 org.hbbtv.ipplayer.setRelativeVolume ... 86
9.9.4.5.6 org.hbbtv.ipplayer.pause ... 87
9.9.4.5.7 org.hbbtv.ipplayer.resume ... 88
9.9.4.5.8 org.hbbtv.ipplayer.seek ... 89
9.9.4.5.9 org.hbbtv.ipplayer.selectComponents ... 89
9.9.4.5.10 org.hbbtv.ipplayer.resolveTimeline ... 90
9.9.5 Regular application control of media presentation ... 91
9.9.6 Resource management considerations ... 94

10 Capabilities ... 95
10.1 Terminal capabilities and functions ... 95
10.1.1 Component selection ... 95
10.1.1.1 Introduction ... 95
10.1.1.2 Component selection via user preferences ... 95
10.1.1.3 Direct component selection via BroadcastSupervisor class ... 96
10.1.1.4 Standard direct component selection ... 96
10.1.1.5 Clarification of component selection by regular HbbTV® applications ... 97
10.1.1.6 Integration between MSE-based media playback and terminal user interface for component selection ... 97
10.1.2 Minimum terminal capabilities ... 98
10.1.3 User Input ... 98
10.1.4 HbbTV® reported capabilities and option strings ... 100

11 Security ... 100
11.1 Overview ... 100
11.2 Device and Server Authentication ... 101
11.2.1 Mutual TLS Authentication ... 101
11.2.1.1 Overview ... 101
11.2.1.2 Client certificate ... 101
11.2.1.2.1 Client certificate overview ... 101
11.2.1.2.2 Operational considerations ... 102
11.2.1.2.3 Client Root and Intermediate Certificate Authority Certificate Profiles ... 102
11.2.1.2.4 Client certificate profile ... 103
11.2.2 Device authentication in broadcast (informative) ... 104
11.3 Operator application authentication ... 104
11.3.1 Encrypted application package overview ... 104
11.3.2 Operator Signing Certificate ... 104
11.3.3 Terminal Packaging Certificate ... 106
11.3.4 Encrypted application packaging process ... 107
11.3.4.1 Encrypted application packaging process overview ... 107
11.3.4.2 Operator application signing process ... 107
11.3.4.3 Process for encrypting an application package ... 108
11.3.4.4 Process for decrypting an application package ... 109
11.3.4.5 Application ZIP package signature verification process ... 109
11.4 CI Plus ... 110
11.4.1 CI Plus communication ... 110
11.4.2 IP Delivery CICAM Player Mode ... 110

12 Privacy ... 111

13 Media synchronization ... 111

14 Companion screens ... 112

15 Communication with regular HbbTV® applications ... 112
15.1 General ... 112

15.2 Service endpoints provided by the terminal ... 112
15.3 Handling of new connections from clients ... 113
15.4 Connection Pairing ... 113
15.5 Paired Connections ... 113

Annex A (normative): OIPF specification profile ... 114

A.1 Detailed section-by-section definition for volume 5 ... 114

A.2 Modifications, extensions and clarifications to OIPF volume 5 ... 121
A.2.1 Configuration class ... 121
A.2.1.1 Constants ... 121
A.2.1.2 Properties ... 123
A.2.1.3 Methods ... 123
A.2.1.4 Replacing UI relating to scheduled recordings ... 124
A.2.1.4.1 Messages ... 124
A.2.1.4.2 Conflict resolution (informative) ... 124
A.2.1.5 Replacing reminders (informative) ... 125
A.2.2 Application class ... 125
A.2.2.1 Properties ... 125
A.2.2.2 Methods ... 126
A.2.2.3 Events ... 130
A.2.3 ChannelConfig class ... 130
A.2.4 Operator applications and the video/broadcast object ... 131
A.2.4.1 Modifications to the state machine ... 131
A.2.4.2 Modification to onChannelChangeSucceeded ... 132
A.2.5 The BroadcastSupervisor class ... 132
A.2.6 oipfDrmAgent ... 137
A.2.6.1 Shared use of DRM and Conditional Access messaging in the terminal ... 137
A.2.7 oipfRecordingScheduler ... 137
A.2.8 Extensions to the application/oipfParentalControlManager object ... 138
A.2.8.1 Properties ... 138
A.2.8.2 Methods ... 138
A.2.8.3 Clarifications ... 138
A.2.9 Extensions to the Channel class ... 139
A.2.9.1 Properties ... 139

Annex B (normative): HbbTV® use of the DVB URI_linkage_descriptor ... 140

B.1 Introduction ... 140

B.2 URI_linkage_descriptor profile ... 140

Annex C (informative): Sequence diagrams ... 141

C.1 Channel Change ... 141

C.2 Broadband-based operator application discovery ... 142

C.3 Broadcast-based operator application discovery and installation ... 142

C.4 Channel change to an operator application presented channel ... 143

Annex D (informative): Bilateral agreement ... 145

Annex E (informative): Bilateral agreement profiles ... 149

E.1 Introduction ... 149

E.2 Profiles ... 149
E.2.1 IPTV Operator profile ... 149
E.2.1.1 Introduction ... 149
E.2.1.2 Quality assurance ... 149
E.2.1.3 Software updates ... 150
E.2.1.4 Branding ... 150
E.2.1.5 Organisation ID ... 150

E.2.1.6 Size of the operator application 150
E.2.1.7 Type of operator application 151
E.2.1.8 Discovery 151
E.2.1.9 Installation of the operator application 151
E.2.1.10 Deciding which operator applications to install 151
E.2.1.11 Failure of the installation of the operator application 151
E.2.1.12 Non-connected operation 151
E.2.1.13 Use and size of the private persistent storage 152
E.2.1.14 Starting the operator application 152
E.2.1.15 Stopping the operator application 152
E.2.1.16 Operator application state change 153
E.2.1.17 Power-on behaviour 153
E.2.1.18 Active standby 153
E.2.1.19 Start pages of the operator application 153
E.2.1.20 Visible RF-Based channel list 153
E.2.1.21 Mandatory API 153
E.2.1.22 Web notifications 153
E.2.1.23 Operator application state change 153
E.2.1.24 Error codes 153
E.2.1.25 Authentication, encryption and decryption of the operator application 154
E.2.1.26 Code and data visiblity 154
E.2.1.27 Replacement and suppression of UI elements 154
E.2.1.28 Key events 154
E.2.1.29 Parental control 155
E.2.1.30 Scheduling recordings 155
E.2.1.31 Channel scan 155
E.2.1.32 Proprietary TV functions 155
E.2.1.33 Performance 156

Annex F (normative): Mapping between HTML5 video element and CICAM player mode 157

F.1 Introduction (informative) 157

F.2 Determining when to use CICAM player mode 159

F.3 Starting CICAM player 159

F.4 During CICAM player use 159

F.5 Stopping CICAM player use 160

F.6 Play to end of content 160

F.7 Errors 161

F.8 Play speed 161

F.9 Random access 161

F.10 Tracks 162

F.11 No mapping possible 162

Annex G (informative): Sequence diagrams for parental access control 162

G.1 Operator Application Triggered Channel Selection for Operator Offered Channels 162

G.2 Operator Application triggered Channel Change non operator offered channel 163

G.3 Terminal Triggered Channel Change for operator offered Channel 164

G.4 Terminal Triggers non Operator Offered Channel Change 164

G.5 Program Change on OpApp Offered Rated Channel 165

G.6 Channel Selection for Locked, non Rated Channel 166

Annex H (normative): Electronic attachments 166

History 168

1 Scope

The present document specifies a platform, based on the existing HbbTV® specification ETSI TS 102 796 [], that supports the signalling, transport and presentation of an operator application. The operator application is able to replace some of the terminal's user interface. The extent to which the terminal user interface is replaced by an operator application depends on the type of the operator application and the business models of the operator and manufacturer. The present document assumes the presence of an agreement between an operator and the device manufacturer. Operator applications will not run in the absence of such an agreement. Topics that could or need to be covered by such a bilateral agreement are listed in annex D.

The present document makes use of functionalities described in ETSI TS 102 796 [] which is describing a platform for signalling, transport, and presentation of enhanced and interactive applications intended for running on hybrid terminals that include both a DVB compliant broadcast connection and a broadband connection to the Internet. The usage of a hybrid terminal for IPTV delivered audio-visual content is described in "IP-delivered Broadcast Channels and Related Signalling of HbbTV Applications" [i.4].

2 References

2.1 Normative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

The following referenced documents are necessary for the application of the present document.

[1] ETSI TS 102 796: "Hybrid Broadcast Broadband TV".

NOTE: The present document is not suitable to be used with versions before 1.4.1.

[2] Open IPTV Forum Release 2 specification, volume 5 (V2.3): "Declarative Application Environment".

NOTE: Available at http://www.oipf.tv/specifications.

[3] ETSI TS 102 809: "Digital Video Broadcasting (DVB); Signalling and carriage of interactive applications and services in Hybrid broadcast/broadband environments".

[4] ETSI TS 102 851: "Digital Video Broadcasting (DVB); Uniform Resource Identifiers (URI) for DVB Systems".

[5] CI Plus™ specification (V1.3.2) (2015-03): "Content Security Extensions to the Common Interface".

NOTE: Available from: http://www.ci-plus.com/data/ci-plus_specification_v1.3.2.pdf.

[6] IETF RFC 2782: "A DNS RR for specifying the location of services (DNS SRV)".

[7] ETSI TS 103 205: "Digital Video Broadcasting (DVB); Extensions to the CI Plus™ Specification".

[8] W3C Recommendation "Web Notifications", 22 October 2015

NOTE: Available at https://www.w3.org/TR/2015/REC-notifications-20151022/

[9] ISO/IEC 21320-1 (2015-10-15): "Information Technology - Document Container File".

NOTE: Available at: http://standards.iso.org/ittf/PubliclyAvailableStandards/c060101_ISO_IEC_21320-1_2015.zip.

[10] IETF RFC 5246: "The Transport Layer Security (TLS) Protocol Version 1.2".

[11] IETF RFC 5280: "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile".

[12] IETF RFC 5652: "Cryptographic Message Syntax (CMS)".

[13] IETF RFC 4055: "Additional Algorithms and Identifiers for RSA Cryptography for use in the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile".

[14] ETSI EN 300 468: "Digital Video Broadcasting (DVB); Specification for Service Information (SI) in DVB systems".

[15] Open IPTV Forum Release 2 specification, volume 7 (V2.3): "Authentication, Content Protection and Service Protection".

NOTE: Available at http://www.oipf.tv/specifications.

[16] IETF RFC 3986: "Uniform Resource Identifier (URI): Generic Syntax".

[17] IETF RFC 4501: "Domain Name System Uniform Resource Identifiers".

[18] IETF RFC 3447: "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1".

[19] W3C Recommendation (28 October 2014): "HTML5 - A vocabulary and associated APIs for HTML and XHTML".

NOTE: Available at http://www.w3.org/TR/2014/REC-html5-20141028/.

[20] W3C Recommendation (19 April 2016): “Web Storage (Second Edition)”

NOTE: Available at https://www.w3.org/TR/2016/REC-webstorage-20160419/#the-storage-interface

[21] W3C Recomendation “Media Source Extensions”

NOTE: Available at https://www.w3.org/TR/media-source/

[22] W3C Editor's Draft (18 January 2020): "Media Playback Quality"
NOTE: Available at https://w3c.github.io/media-playback-quality/

[23] "Streaming Quality of Experience Events, Properties and Metrics (CTA-2066)"

[24] ETSI TS 102 809: “Digital Video Broadcasting (DVB); Signalling and carriage of interactive applications and services in Hybrid broadcast/broadband environments”.

[25] WHATWG Notfications API Review Draft – Published 23 July 2018

NOTE: Available at https://notifications.spec.whatwg.org/review-drafts/2018-07/

[26] ETSI TS 102 796: "Hybrid Broadcast Broadband TV" version 1.7.1.

[27] Phillips, A., Ed., and M. Davis, Ed., "Tags for Identifying Languages", BCP 47, RFC 5646, September 2009.

NOTE: Available at https://tools.ietf.org/rfc/bcp/bcp47.txt

[28] ETSI TS 102 034: “Digital Video Broadcasting (DVB); Transport of MPEG-2 TS Based DVB

Services over IP Based Networks”

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] ETSI TS 103 464: "Hybrid Broadcast Broadband TV Application Discovery over Broadband".

[i.2] Open IPTV Forum Release 2.3 specification volume 5a (V2.3): "Web Standards TV Profile".

[i.3] W3C Candidate Recommendation: "Secure Contexts".

[i.4] HbbTV®: "IP-delivered Broadcast Channels and Related Signalling of HbbTV Applications".

[i.5] IETF RFC 6454: “The Web Origin Concept”.

[i.6] DIAL Discovery and Launch Protocol Specification

NOTE: Available at www.dial-multiscreen.org/dial-protocol-specification

[i.7] IETF RFC 3986: "Uniform Resource Identifier (URI): Generic Syntax".

[i.8] W3C Proposed Candidate Recommendation (18th May 2021): “Page Visibility Level 2”

NOTE: Available at https://www.w3.org/TR/2021/CR-page-visibility-2-20210518/

[i.9] W3C® Candidate Recommendation (08 October 2015): "Mixed Content".

NOTE: Available at https://www.w3.org/TR/mixed-content/.

[i.10] IETF, Mike West. Let 'localhost' be localhost.

NOTE: Available at https://tools.ietf.org/html/draft-west-let-localhost-be-localhost.

[i.11] HbbTV Association: “HbbTV Specification 2.0.3”

Note: Available at https://www.hbbtv.org/wp-content/uploads/2020/10/HbbTV-SPEC-00525-HbbTV-SPEC-00515-008-hbbtv203_2020_10_14.pdf

[i.12] Fullscreen API. WHATWG Review Draft – Published 18 January 2021.

NOTE: Available at https://fullscreen.spec.whatwg.org/review-drafts/2021-01/

[i.13] ETSI EN 303 560: “Digital Video Broadcasting (DVB); TTML subtitling systems”.

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the following terms and definitions apply:

bilateral agreement: agreement between a terminal manufacturer and an operator defining commercial, operational, technical and user interface arrangements for the use of an operator application

broadband: bi-directional IP connection with sufficient bandwidth for streaming or downloading A/V content

broadcast: uni-directional MPEG-2 transport stream based broadcast using DVB technologies

companion screen device: device (not another HbbTV® terminal) that can run applications that in turn link to or work with an HbbTV® terminal or HbbTV® application

NOTE: Such a device can be for example a mobile phone or a tablet.

hybrid terminal: terminal supporting delivery of A/V content both via broadband and broadcast

operator: entity that aggregates a set of channels and offers them to the user

operator application: application from an operator that takes over some of the user interface of the terminal

operator application created channel: content that is offered to the consumer in the operator application UI in a way that looks like a TV channel but where the terminal has no independent knowledge of it

operator application presented channel: channels for which streamed media presentation is handled by an operator application using an HTML5 media element

operator-specific operator application: operator application that is installed on a terminal and that, when it is active, provides most of the terminal's user interfaces

NOTE: This type of operator applications is intended for set-top-boxes where the manufacturer provides little or no user interface except perhaps for the basic device setup and installation.

privileged operator application: operator application that is installed on a terminal, can be activated by the user and, when it is active, replaces some of the terminal's user interface

NOTE: This type of operator applications is intended for TV sets.

regular HbbTV® application: HbbTV® application that uses the features defined in ETSI TS 102 796 [] and nothing from the present document except for features specifically identified as being available to regular HbbTV® applications

NOTE: Clause 4.3 identifies features in the present document that are specifically identified as being available to regular HbbTV® applications.

standard operator application: application providing operator functionality using only the features defined in ETSI TS 102 796 []

NOTE: A standard operator application is also a regular HbbTV® application.

terminal: HbbTV® terminal (as defined in ETSI TS 102 796 []) which also supports the detection, installation and execution of operator applications as defined in the present document

3.2 Abbreviations

For the purposes of the present document, the following abbreviations apply:

AIT	Application Information Table
APDU	Application Protocol Data Unit
API	Application Programme Interface
A/V	Audio/Video
ASCII	American Standard Code for Information Interchange
BAT	Bouquet Association Table
CA	Certificate Authority
CSP	Content and Service Protection
CMS	Cryptographic Message Syntax
CI	Common Interface
CICAM	Common Interface Conditional Access Module
CRL	Certificate Revocation List
DAE	Declarative Application Environment
DASH	Dynamic Adaptive Streaming over HTTP
DER	Distinguished Encoding Rules
DNS	Domain Name Service

DOM	Document Object Model
DRM	Digital Rights Management
DSM-CC	Digital Storage Media - Command and Control
DVB	Digital Video Broadcasting
DVB-SI	Digital Video Broadcasting - Service Information
DVB-C	Digital Video Broadcasting - Cable
DVB-S	Digital Video Broadcasting - Satellite
EPG	Electronic Programme Guide
FDP	File Delivery Protocol
FQDN	Fully Qualified Domain Name
HDMI	High Definition Multimedia Interface
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol - Secure
IP	Internet Protocol
IPTV	Internet Protocol TeleVision
MMI	Man Machine Interface
MPEG	Motion Picture Experts Group
NIT	Network Information Table
OIPF	Open IPTV Forum
PVR	Personal Video Recorder
PKI	Public Key Infrastructure
RF	Radio Frequency
RSA	Rivest, Shamir and Adleman
SAS	Specific Application Support
SDT	Service Description Table
SPTS	Single Program Transport Stream
SRV	Service Record
TLS	Transport Layer Security
TV	TeleVision
TXT	Teletext
UI	User Interface
URI	Uniform Resource Identifier
VoD	Video on Demand
XML	eXtensible Markup Language

4 Overview

4.1 Operator applications (informative)

4.1.1 Scope and motivation

Operator applications replace parts or all of the user interface that is otherwise provided by the Terminal. Typically, this includes UI elements of TV watching mode (e.g. channel info banner and channel selection) and the electronic program guide (EPG).

Using operator applications allows for:

- Implementation of a homogenous user experience over all services of the operator.
- Integration of the operator's enhanced services with the basic UI elements.
- Branding of all UI elements.
- Update and extension of the operator's UI without the need of an update of the system software.

The actual set of UI elements provided by a specific operator application depends on its type and on the product design of manufacturer and operator.

4.1.2 Types of operator applications

There are three types of operator applications as shown in table 1.

Table 1: Types of Operator Applications

Standard	Standard operator applications are HbbTV® applications as defined in ETSI TS 102 796 []. They do not replace UI elements of the terminal but may provide alternatives to some of them. The present document does not define any technical extensions for standard operator applications.
Privileged	Privileged operator applications are installed on a terminal. When active, they replace some of the user interface of the terminal. Privileged operator applications are intended for TV sets.
Operator-specific	Operator-specific operator applications are installed on a terminal. When active, they replace virtually all of the user interface of the terminal. Operator-specific operator applications are intended for set-top-boxes where the manufacturer provides little or no user interface.

A terminal may provide support for only privileged operator applications or only operator-specific operator applications or both.

4.1.3 Standard operator applications

4.1.3.1 Introduction

To a certain degree it is possible to use regular HbbTV® applications to implement typical features provided by an operator. Such applications (referred to as standard operator applications) are usually broadcast-related, and they are signalled on all channels of the operator. The following clauses describe the user experience of standard operator applications compared to actual operator applications as defined in the present document (i.e. privileged and operator-specific operator applications).

4.1.3.2 Features of standard operator applications

Standard operator applications cannot replace UI elements of the terminal but may provide alternatives to some of them such as:

- EPG.
- Channel list.
- List of other applications of the operator.
- Promotion of the operator and its content.

4.1.3.3 Design policy for standard operator applications

The design of a standard operator application is solely under the responsibility of the operator.

4.1.3.4 TV channels

A standard operator application is usually only available on channels of the operator. As soon as the user selects a channel that is not offered by the operator, the standard operator application is terminated.

The user may set up a favourite channel list including all channels of the operator. In that case, zapping through the favourite channel list would not change the user experience.

Furthermore, all partners in a network may agree that a standard operator application is allowed to run on all channels of the network.

4.1.3.5 Activating and launching of standard operator applications

It is neither possible nor required for the user to activate a standard operator application before it is actually launched.

The operator and the broadcaster decide how the operator's standard operator application is launched. It may be one of the following:

- The operator application is launched automatically as soon as a channel of the operator has been selected (i.e. the operator application is the autostart application).
- The operator application can be launched by pressing the TXT button of the remote control (i.e. the operator application is signalled as Digital Teletext application).
- The operator application can be launched from an autostart application by pressing a dedicated button on the remote control (e.g. the "green" button).

4.1.3.6 User input for standard operator applications

Standard operator applications can only use key events of buttons as defined for regular HbbTV® applications.

4.1.4 Privileged operator applications

A privileged operator application replaces some parts of the user interface of the terminal. Typically, that includes content related UI elements in TV watching mode such as:

- Channel info banner.
- Channel selection:
  - Channel list.
  - Channel number input.
  - Zapping.
- Component selection:
  - Audio stream selection.
  - Subtitle stream selection.
- Parental control for broadcast content.
- Timeshift control.
- Recording in TV watching mode.
- Messages in regards to programming (e.g. Reminders or Recordings).

It may also include general UI elements in TV watching mode:

- Volume control banner.
- System messages (e.g. "Signal lost", "Upgrade available", "Channel list updated" etc.).
- CA and DRM messages.
- CICAM MMI.

Furthermore, a privileged operator application may replace the UI of the following other embedded applications:

- EPG.
- PVR archive/Media library.

The user interface of the terminal usually covers at least:

- First installation.
- Set-up menu.
- Menu for broadcast-independent HbbTV® applications.
- Access to other sources.

4.1.5 Operator-specific operator applications

An operator-specific operator application can implement all features of privileged operator applications but it can also replace further parts or even all of the user interface of the terminal. It can provide a user experience comparable to that of set-top boxes of PayTV providers.

4.1.6 Coexistence between multiple types of operator application

Standard operator applications, privileged operator applications and operator-specific operator applications may be mixed in a single market or deployment. This could enable a baseline user experience for consumers using a standard HbbTV® terminal and an enhanced user experience for consumers using an implementation of the present document. A standard operator application may be able to detect that a privileged or operator-specific operator application is already running and modify its behaviour. For example:

- In a broadcast service with no broadcast-related HbbTV® applications of its own, a standard operator application could choose to not consume any key events and not show any UI to the user if it detected that an equivalent privileged or operator-specific operator application was already running.
- In a broadcast service with an autostart launcher or menu application, the launcher could modify its UI depending on whether a particular privileged or operator-specific operator application was running. For example, if the operator application was not running it would claim the green key event and start a standard operator application if the green key was pressed. Alternatively, if the operator application was running then it would not claim the green key event and allow that key event to fall through to the operator application.

4.1.7 Operator applications and channel binding

In contrast to the concept for channel binding of regular HbbTV® broadcast-related applications as defined in ETSI TS 102 796 [], the present document does not define any technical means to bind an operator application to a group of channels. It is the operator's responsibility to ensure that no legal or business rules are broken by using the operator application on a channel that is not offered by the operator. Typical solutions for that are introduced in clause 5.5.1.

4.1.8 Enabling operator applications to be installed and run

The present document does not require terminals to automatically install every privileged or operator-specific operator application which is signalled on the network or which is otherwise detected. An operator cannot expect that its operator application automatically runs on every terminal even if the terminal in principle is capable of running operator applications. Instead, the present document requires privileged and operator-specific operator applications to be authenticated before being run. It is expected that this authentication forms part of a bilateral agreement between the operator and the terminal manufacturer. Operator application authentication is introduced in clause 11.3 of the present document. Some details of the operator application authentication as well as further content of the bilateral agreements are out of scope of the present document. However, Annex D provides a list of possible topics which could be covered.

4.1.9 Number of operator applications

A terminal may support only one or several privileged or operator-specific operator applications. In any case only one of these operator applications can run at a certain time.

If several operator applications are supported and available, the terminal provides a method to select one, usually using one of the following concepts:

1) The user selects the operator application of their choice (e.g. as part of the first installation dialogue or via a source selection menu).

2) The terminal itself selects the most appropriate operator application (e.g. based on network, location or user data).

4.2 Architecture (informative)

4.2.1 Introduction

This clause gives an overview of a system architecture of a terminal. The architecture which allows for the provision of operator applications comprises a browser, application signalling via broadcast, broadband and from a companion device, application transport via broadcast, broadband and from a CICAM, and synchronization of applications and broadcast services.

4.2.2 System overview

Regarding the usage of operator applications a terminal has the capability to be connected to two networks in parallel. On one side it can be connected to a broadcast DVB network. Via this broadcast connection the terminal can receive operator applications as well as regular HbbTV® applications. Due to the uni-directional characteristics of a broadcast network the terminal will not be able to communicate via this connection with the application providers. A bi-directional communication however is possible if the terminal is connected to the Internet via a broadband interface. Via both interfaces the terminal can receive both regular HbbTV® applications and operator applications. Both types of applications can be active in the terminal simultaneously. Additionally, both interfaces can also be used for conveying standard broadcast A/V. Non-realtime A/V content can be either transmitted via broadcast (using the FDP protocol), or via the broadband interface that may also connect to companion screen devices.Figure 1 depicts the system overview for such a terminal considering the transport of regular HbbTV® applications and operator applications with DVB-S as the example of the broadcast connection.

4.3 Referenced W3C and WHATWG Specifications

Clause 4.5 of TS 102 796 [] concerning references to W3C and WHATWG specifications shall also apply to W3C and WHATWG specifications referenced from the present document.

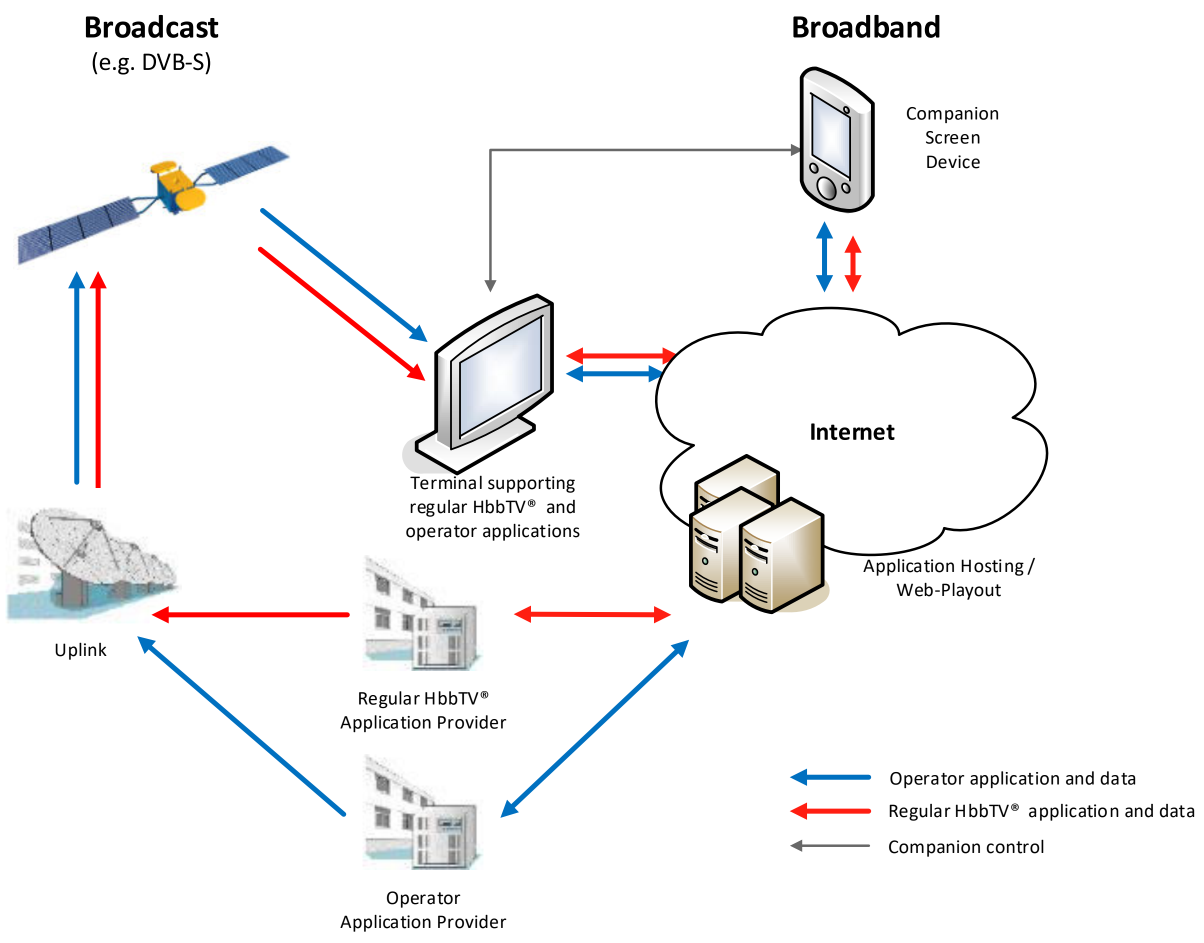
Broadcast
(e.g. DVB-S)
Broadband
Companion Screen Device
Internet
Terminal supporting regular HbbTV® and operator applications
Application Hosting / Web-Playout
Uplink
Regular HbbTV® Application Provider
Operator Application Provider
Operator application and data
Regular HbbTV® application and data
Companion control

Figure 1: System overview for an operator application terminal

4.3 Dependencies

Operator applications shall be able to successfully call all APIs and successfully use all features and functions that a regular HbbTV application on the same terminal is able to use, unless otherwise stated in the present document. For example, the ability to present broadcast and broadband delivered video and audio and synchronise them and the ability to interact with a companion screen can all be used by Operator applications.

For a regular HbbTV® application, the functionality provided by the terminal is altered as described in the present document. Table 1 indicates the clauses that describe additions or changes to the APIs available to a regular HbbTV® application.

Table 1: Clauses that define functionality for regular HbbTV applications

Clause	Functionality for regular HbbTV® applications defined in the present document
8.4.1	Web Notifications API availability.
9.9.6	Resource management considerations
A.2.1	Configuration class: • `runningOperatorApplication` property
A.2.2	Application class: • `opAppState` property • `createApplication()` method • `getApp2OpAppBaseUrl()` method

5 User experience (informative)

5.1 Introduction

This clause describes the usage of operator applications as seen by the end-user. It should be considered as guidelines for implementing operator applications and for setting up the bilateral agreements between manufacturer and operator.

5.2 Using operator applications

5.2.1 Activation and deactivation

The terminal provides means to activate an operator application. The provided method may depend on the following:

- Types of supported operator applications
- Number of supported operator applications
- Requirements of the target markets
- User interaction paradigms of the terminal

Table 2 introduces some typical methods to activate an operator application.

Table 2: Operator application activation

Method	Description
Permanent activation	A specific operator application is always activated when the terminal is running. • This method may be the chosen one for a terminal of a PayTV provider. • This method does not require any user interaction.
Activation during first installation	The user selects an operator application during the first installation process. • This method may be the chosen one for a terminal which supports operator-specific operator applications of several operators. • The user may have to perform a factory reset to deactivate the operator application or to select a different operator application.
Selection as source	The terminal adds an entry for an operator application in its source selection menu, which usually provides entries such as "TV", "Radio", "HDMI", "PVR" or "Apps". • This method may be the chosen one for a terminal which supports one specific privileged operator application. • If the user selects a different source, the operator application is deactivated by the terminal. The user then has to re-select the operator application to activate it again.
Combination of set-up and source selection	The set-up menu of the terminal provides a dialogue to select an operator application. The terminal automatically activates the selected operator application whenever the user selects a source providing content of the operator (e.g. "TV" or "Radio"). • This method may be the chosen one for a terminal which supports privileged operator applications of several operators. • To deactivate the operator application or to select a different operator application, the user can use the set-up menu, too.
Operator application key (e.g. EPG, Channel list) or a dedicated operator key	Some of the keys defined in clause 10.1.3 as being "operator application key"s may result in the operator application being activated (started) when it is not already running. Remote controls may also include a dedicated operator key.
Selection of channel from terminal native UI	The terminal changes to the selected channel and subsequently starts the operator application with the opapp-select-channel startup location (see Table 11). The operator application can determine the selected channel with property `BroadcastSupervisor.currentChannel`.

5.2.2 User input

The user controls operator applications using a user input device typically supplied with the terminal. This may be a conventional remote control or an alternative input device.

Table 3 describes the buttons or key events that are relevant for the end user when using operator applications. Which of these buttons are actually used and how they are used depends on the operator application and the bilateral agreement between manufacturer and operator.

Table 3: Operator application key events

Buttons	Typical usages
4 colour buttons 4 arrow buttons ENTER/OK BACK 10 number buttons	Application specific
2 program selection buttons	Zapping through the channel list as managed by the operator application
subtitle button	Activate or deactivate subtitles Open dialogue for subtitle selection
audio description button	Activate or deactivate audio description Open dialogue for selection of audio description
audio track button	Open dialogue for selection of audio stream
play stop pause fast forward fast rewind	Manage Timeshift Playback
record	Start live recording Program future recording in EPG
GUIDE or EPG	Open EPG of the operator
INFO	Display channel info banner
channel list button	Display channel list
2 volume control buttons mute	Adjust volume Display volume info banner
EXIT	If video is playing, hide the operator application UI. If no video is playing, return to a higher level UI or home screen.

Somebehaviour of the EXIT button is defined in the present document and implemented by the manufacturer. Otherwise, the behaviour of all other buttons may be implemented by the operator application in accordance with the bilateral agreement.

An operator-specific operator application may use further buttons or even all buttons that are available on the supported interaction devices.

5.3 Displaying operator applications

5.3.1 General visibility of operator applications

If an operator application is activated, it is not automatically visible to the user. Instead, the operator application can become visible in cases like the following:

- The operator application has been selected as a source.
- The user has left an application of the terminal (e.g. Set-up or HbbTV® application menu) and returns to TV.
- A new TV channel has been selected (by the user or by the operator application or by the terminal).
- The user has pushed a button on the remote control (e.g. the INFO or GUIDE button).
- A notification needs to be shown.

If a TV channel is shown in the background, the operator application should automatically hide itself after a time-out. If a Radio channel is running or if no broadcast channel is running at all, the operator application may stay visible.

NOTE: Depending on the version of TS 102 796 [1] supported and the HTML5 engine used in the implementation, terminals may support the Page Visibility API [i.8] (or an earlier or later version). Operator applications may register to receive visibilitychange events however this is not recommended as these overlap with OperatorApplicationStateChange events as defined in the present document and, unlike OperatorApplicationStateChange events, visibilitychange events are not tested by the unit tests for the present document so their behaviour may not be as reliable or consistent.

5.3.2 Start page

The operator application may have different start pages depending on the launch and/or startup contexts. For example:

- Selecting the operator application in the source selection menu may lead to decoding a TV channel and showing the channel info banner of the operator application.
- Pressing the GUIDE button while using an application of the terminal may lead to the operator application showing its EPG.

5.3.3 Co-existence of operator application and regular HbbTV® applications

If a regular HbbTV® application is launched while an operator application is visible, the operator application usually becomes invisible to avoid any confusion (see clause 6.5.2). However, the operator application is not terminated but keeps running in the background. If the launched regular HbbTV® application is terminated, the calling operator application may be displayed again if that behaviour has been requested by the operator application.

If an operator application becomes visible while a regular HbbTV® application is running, parts of the regular HbbTV® application stay visible if the operator application does not cover the whole screen. A terminal may indicate this change to the user, perhaps by making any UI from the regular HbbTV® application darkened or faded to a certain degree.

5.3.4 Co-existence of operator application and terminal UI

If the operator application does not provide all UI elements of TV watching mode, the UI of the operator application may be overlaid with UI of the terminal in certain cases. Typical examples are system messages or the volume control banner or a parental control dialogue. If the operator application provides all necessary UI elements, such a mix of user interfaces of different sources can be avoided.

5.4 Design policy

5.4.1 Branding

Usually, an operator application is designed and branded by the operator in agreement with the manufacturer.

Due to market specific regulations it may be necessary to offer the operator application with a branded and a neutral design for channels of the operator and other channels, respectively.

5.4.2 User interface design constraints

If broadcast content is running, the display of graphics by privileged operator applications is limited to the following:

- UI responses to user input for up to 60s using the Transient state defined in clause 6.3.3.4.
- Display of a channel information banner when the channel is changed, also using the Transient state.
- Notifications that relate to a previous user action using the mechanism defined in clause 8.4.1.1 and following the guidelines in clause 8.4.1.2. These are displayed by the terminal on behalf of the operator application.
- No limitations apply when broadcast video is hidden or scaled so that both width and height are not bigger than 1/3 of the screen size (see the Foreground state defined in clause 6.3.3.2).

There are no limitations in placed on the design of operator-specific operator applications apart from market specific laws and regulations.

5.5 Further concepts

5.5.1 Channels not offered by the operator

Central elements of operator applications such as the channel info banner are usually displayed in parallel or on top of a TV channel. Due to legal or business requirements it might be necessary to prevent the operator application from showing its user interface on channels not offered by the operator. There are four typical solutions for that listed in table 4.

Table 4: Channels not offered by an operator

Solution	Description
Restricted channel list	The channel list (and any other channel selection method) of the operator application covers only channels offered by the operator. In that case the user has to deactivate the operator application before being able to select channels not offered by the operator.
Self-deactivation	The operator application provides access to channels not offered by the operator, but it terminates itself as soon as the user selects such a channel. In that case the terminal automatically provides its standard user interface on channels not offered by the operator. The user has to reactivate the operator application before using it again.
Two different designs	The operator application provides access to channels not offered by the operator, but it uses a different non-branded design if such a channel is selected. If the user selects a channel offered by the operator, the operator application automatically uses again its actual (branded) design. This solution does not require any further user interaction. The interpretation of "non-branded" depends on local regulations.
Terminal-driven solution	The terminal itself prevents the operator application from showing its user interface on channels not offered by the operator. For this the terminal may have to manage a list of channels of the operator. Selection of a different channel may lead to termination of the operator application. Selection of a channel offered by the operator may lead to relaunching the operator application. Such a solution has to be defined in the bilateral agreement between manufacturer and operator.

6 Service and application model

6.1 Operator application discovery and installation

6.1.1 Overview

The following clause describes the process the terminal shall perform to discover, retrieve and install an operator application. The present document allows deployment scenarios both with and without an operational broadband connection.

In both scenarios, an operator application is described by an AIT (either in binary or XML encoding).

NOTE: In the following clauses, a reference to "AIT" always means the binary encoding. A reference to "XML AIT" always means the XML encoding. A reference to "(XML) AIT" applies to either encoding.

Once the location of the (XML) AIT is discovered, the terminal downloads it and starts the operator application retrieval process.

The availability of an operator application may be signalled within the operator's broadcast or broadband network(s). Within broadcast networks, the operator application is signalled via descriptors within the operator's NIT and/or BAT. Within broadband networks, the operator application is signalled via standard DNS methods. Alternatively, a terminal may be provisioned with a hardwired location of the XML AIT.

The terminal shall follow a three step process to complete the operator application discovery and retrieval process:

1) The terminal shall perform an (XML) AIT location discovery as defined in clause 6.1.2.

2) The terminal shall perform an (XML) AIT acquisition as defined in clause 6.1.5.

3) The terminal shall download the encrypted application package as defined in clause 6.1.7.

The terminal shall not attempt the retrieval of any data via broadband if the terminal does not have an operational broadband connection.

Table 5 below defines the 7 combinations of (XML) AIT discovery, download and application package download mechanisms that are included in the present document. Other combinations are not included.

Table 5: Combinations of (XML) AIT discovery, download and application package download methods

	Method	Reference	Combination						
			1	2	3	4	5	6	7
(XML) AIT discovery	Broadcast NIT/BAT with URI_linkage_descriptor with operator FQDN	6.1.3.2	✓						
	Broadcast NIT/BAT with URI_linkage_descriptor with URI of AIT	6.1.3.3						✓	
	NIT from CICAM with uri_linkage_descriptor with operator FQDN	Not included in the present document							
	NIT from CICAM with uri_linkage_descriptor with URI of XML AIT	6.1.3.4					✓		✓
	Hardwired in terminal with operator FQDN	6.1.3.5		✓					
	Hardwired in terminal with URI of XML AIT	6.1.3.6			✓				
	DNS SRV lookup to a standardised address	6.1.3.7				✓			
(XML) AIT download	XML AIT from broadband	6.1.5.1	✓	✓	✓	✓	✓		
	AIT from broadcast	6.1.5.2						✓	
	XML AIT from broadcast carousel	Not included in the present document							

	XML AIT from CICAM auxiliary file system	6.1.7.4							✓
Operator application package download	Via IP	6.1.7.2	✓	✓	✓	✓	✓		
	Via DSM-CC object carousel	6.1.7.3						✓	
	From CICAM auxiliary file system	6.1.7.4							✓

There may be other methods of installing an operator application that do not rely on (automatic) discovery, subject to the bilateral agreement between Operator and TV manufacturer. A few examples:

- The application is pre-installed on a device that is sold in a certain market. This application may have been assigned a 'source' already, or it may guide the user through an installation process the first time it is opened, after which it is installed as a 'source'. A pre-installed privileged operator application shall only run with a user's prior consent, and it shall be possible for the user to uninstall the application (see clauses 6.7 and 12).
- The application is made available in the manufacturer's app store. When a user installs the app (for example, following instructions provided by the operator), and subject to the user agreeing, the app is installed as a 'source'.
- During the initial installation by the user, the application is selected by that user from a list of operator applications that are available in a specific market.

More than one operator application may be available in a certain market; clause 6.1.6 provides some suggestions on how this can be handled.

6.1.2 Triggering operator application discovery

The process of operator application discovery shall be triggered by the conditions listed in table 6.

Table 6: Triggering operator application discovery process

Trigger	Scope of resulting discovery
Terminal is being installed and there is at least one applicable operator application (see note 1).	All discovery methods used by all applicable operator applications.
As above but the terminal is re-installed or a factory reset is applied.	
Terminal UI provides a mechanism to enable the user to trigger the discovery process.	
The user installs a CICAM and at least one applicable operator application uses the CICAM discovery method.	As defined in clause 6.1.3.4.
The manufacturer deploys a software update to the terminal that adds at least one applicable operator application.	Either 1) the discovery methods used by the added applicable operator applications or 2) all discovery methods used by all applicable operator applications.
The manufacturer adds at least one applicable operator application via some kind of secure provisioning mechanism.	
At least one applicable operator application uses discovery based on a URI_linkage_descriptor in a NIT/BAT and the NIT/BAT specified in the bilateral agreement for those operator application(s) is changed, e.g. to add a uri_linkage_descriptor where one was not present before (see note 2).	As defined in clauses 6.1.3.2 and 6.1.3.3.
NOTE 1: An "applicable operator application" as used in this table means one from an operator that is in the terminal's list of supported operators (see clause 6.6.1) and which is deployed in the country or market where the terminal is installed. NOTE 2: As a consequence, terminals supporting discovery based on the uri_linkage_descriptor need to monitor for changes. The exact details of this need to be defined in the bilateral agreement along with the location of the NIT/BAT concerned and whether any NIT is a NIT actual_network or a NIT other_network as defined in ETSI EN 300 468 [14], clause 5.2.1 and table 2.	

6.1.3 Operator application discovery methods

6.1.3.1 Introduction

The terminal discovers and retrieves the location of the (XML) AIT as represented by a URI. The URI shall adhere to rules specified in clause 9.2 of ETSI TS 102 796 [].

The location of the (XML) AIT is signalled by the operator using one or more of the following methods:

1) Broadcast NIT/BAT containing a URI_linkage_descriptor with the operator FQDN (requiring DNS SRV look up to get the URI of the XML AIT as defined in clause 6.1.4).

2) Broadcast NIT/BAT containing a URI_linkage_descriptor with the URI of the AIT.

3) NIT from CICAM containing a URI_linkage_descriptor with the URI of the XML AIT.

4) Operator FQDN hardwired in the terminal (requiring DNS SRV look up to get the URI of the XML AIT as defined in clause 6.1.4).

5) URI of the XML AIT hardwired in the terminal.

6) DNS SRV lookup to a standardized address (as defined in clause 6.1.4).

Each of these is described in more detail below.

6.1.3.2 Broadcast NIT/BAT with URI_linkage_descriptor with operator FQDN

The terminal shall parse the first loop of the broadcast NIT/BAT used for service discovery to locate the URI_linkage_descriptors matching the profile in Annex B and with a scheme of "dns" in the uri_char. If any are found, the terminal shall extract the FQDN from the uri_char of each of them, which should each be a DNS URI as defined in IETF RFC 4501 [17] without any "dnsauthority" or "dnsquery" parts (e.g. dns:foo.example.com). The terminal shall take the "dnsname" part of each URI and use that FQDN to perform the process defined in clause 6.1.4.

NOTE: Exactly which broadcast NIT/BAT is used for service discovery is outside the scope of the present document.

6.1.3.3 Broadcast NIT/BAT with URI_linkage_descriptor with URI of AIT

The terminal shall parse the first loop of the broadcast NIT/BAT used for service discovery to locate the URI_linkage_descriptors matching the profile in Annex B and with a scheme of "dvb" in the uri_char. If any are found, the terminal shall extract the DVB URI from the uri_char of each of them and perform the process defined in clause 6.1.5.2.
The uri_char shall contain a DVB URI conforming to clause 6 of ETSI TS 102 851 [4] containing only original network id, service id and optionally transport stream id but no other part.

NOTE: Exactly which broadcast NIT/BAT is used for service discovery is outside the scope of the present document.

6.1.3.4 NIT from CICAM with uri_linkage_descriptor with URI of XML AIT

If the terminal supports Operator Profile (as specified in CI Plus Specification v1.3.2 [5]) and the CICAM reports a profile_type of 1, the terminal shall parse the first loop of the NIT from the CICAM to locate the URI_linkage_descriptors matching the profile in Annex B and with a scheme of either "https" or “ci://” in the uri_char. If any are found, the terminal shall extract the URI from the uri_char of each of them and perform the process defined in clause 6.1.5.1. URI_linkage_descriptors with a scheme of "http" in the uri_char shall be ignored.

When this discovery method is run following a request to update an installed OpApp (see clause 6.2), the most recently previously installed or updated CICAM NIT shall be used.

NOTE: The present document does not define the use of profile_type 2 (as specified in CI Plus Specification v1.4 [7]). In particular, how to merge different URI_linkage_descriptors coming from broadcast and the CICAM is out of scope of the present document.

6.1.3.5 Hardwired in terminal with operator FQDN

If the terminal is provisioned with an operator FQDN appropriate for the country for which the terminal is configured, then the terminal shall perform the process defined in clause 6.1.4.

NOTE: The bilateral agreement where the operator FQDN is specified may define which countries are appropriate for its use.

6.1.3.6 Hardwired in terminal with URI of XML AIT

If the terminal is provisioned with URI to an XML AIT appropriate for the country for which the terminal is configured, then the terminal shall perform the process defined in clause 6.1.5.1. This shall be an "https://" URI.

NOTE: The bilateral agreement where the operator XML AIT is specified may define which countries are appropriate for its use.

6.1.3.7 DNS SRV lookup to a standardized address

The terminal shall use the FQDN "hbbtvopapps.org" and shall perform the process defined in clause 6.1.4.

NOTE: This mechanism is intended for use only within an operator's IP network and would likely not work in the open internet. This FQDN is deliberately different from that used in ETSI TS 103 464 [i.1] which will be used by broadcasters in the open internet. Use of custom DNS servers by either a terminal or a router will likely result in the failure of this mechanism. This mechanism should only be relied upon in deployments where this failure will not happen.

6.1.4 DNS SRV lookup process

The terminals shall use the FQDN discovered in clauses 6.1.3.2 and 6.1.3.5 to construct a DNS SRV lookup as specified in IETF RFC 2782 [6] by prefixing the FQDN with the string "_hbbtv-ait._tcp.".

EXAMPLE: Assuming the FQDN example.com , the following could be the SRV request and response:

Request:

_hbbtv-ait._tcp.example.com

Response:

_hbbtv-ait._tcp.example.com 3600 IN SRV 10 1 443 ait.example.com.

The terminal shall use the host name and port number from the SRV lookup to construct the URL for retrieving the XML AIT, appending "/opapp.aitx" and prefixing it with "https://", resulting in a URL of the form https://ait.example.com:443/opapp.aitx.

The terminal shall use this URL with the process defined in clause 6.1.5.1.

NOTE: RFC 2782 [6] says, "The domain name of the target host ... MUST NOT be an alias (in the sense of RFC 1034 or RFC 2181)". This conflicts with normal CDN usage and is widely ignored in the real world. Terminal support for behaviour outside the RFC cannot be assumed.

6.1.5 (XML) AIT acquisition and download

6.1.5.1 XML AIT acquisition

If the terminal discovers the location of an XML AIT using DNS SRV as defined in clause 6.1.4, the terminal shall perform a HTTP GET request based on the priority and weighting of the returned SRV records as specified in IETF RFC 2782 [6]. If an HTTP GET request for one XML AIT fails, the terminal shall perform another HTTP GET request using other returned SRV records in order of priority and weighting as specified in IETF RFC 2782 [6].

If the terminal discovers the location of one or more XML AIT with a scheme of “https” as defined in clause 6.1.3, the terminal shall perform HTTP GET requests using all those https:// URLs.

TLS shall be used as defined in clause 11.2.1 of ETSI TS 102 796 [] with mutual authentication as defined in clause 11.2.1 of the present document.

Terminals shall provide the HTTP User-Agent as defined in clause 7.3.2.4 of ETSI TS 102 796 [1] when requesting the XML AIT file. Terminals shall also respect the Cache-Control HTTP response header to cache and not re-fetch the XML AIT file while it is valid according to these headers.

Servers shall respond correctly to the If-Modified-Since HTTP request header when the terminal requests the XML AIT.

Terminals shall append the additional parameters set by the operator application as defined in A.2.2.2. An example URI constructed by the terminal when an operator application sets the parameter argument as “?version=1” results in the following constructed URL “https://ait.example.com:443/opapp.aitx?version=1”.

If the terminal discovers the location of one or more XML AIT with a scheme of “ci://” as defined in clause 6.1.3.4 and the terminal supports the discovery of an OpApp from the CI Plus Auxiliary File System then the terminal shall obtain the XML AIT by use of the CI Plus Auxiliary File System. The URL schemes for the HbbTV Application accessing files are as defined in the clause 9.2 of HbbTV Specification.

The terminal shall parse the XML AIT according to the requirements defined in clause 7.2.3.2 of ETSI TS 102 796 [] with the additional requirements in table 7 and clause 7.3.1 of the present document.

Table 7: XML AIT Profile

Field or element	Requirement on XML AIT file	Requirement on terminal
`applicationUsageDescriptor/ApplicationUsage`	Shall be "`urn:hbbtv:opapp:privileged:2017`" for privileged operator applications or "`urn:hbbtv:opapp:opspecific:2017`" for operator specific operator applications.	Mandatory for terminal to use. XML AITs with other values shall be ignored.
`applicationDescriptor/version`	Mandatory. It shall increment whenever the encrypted application package (as defined in clause 11.3) changes. Values shall never be reused.	Mandatory. XML AITs without a version shall be ignored.
`applicationDescriptor/type/OtherApp`	Shall be "`application/vnd.hbbtv.opapp.pkg`".	Mandatory. XML AITs with other values shall be ignored.
`applicationTransport/`	The applicationTransport field in the XML AIT shall be either HTTPTransportType (as defined in TS 102 809 [24]) or CITransportType (as defined in TS 103 205 [7])	Mandatory for terminal to use. XML AITs with other values shall be ignored.

The present document specifies the XML AIT schema in ETSI TS 102 796 [] for the "icon" element to be interpreted as per clause 5.2.8.3 of ETSI TS 102 796 [] which allows "one or more IconDescriptor elements", i.e. in the XML AIT schema this line "<xsd:element name="icon" type="mhp:IconDescriptor" minOccurs="0"/>" is changed to "<xsd:element name="icon" type="mhp:IconDescriptor" minOccurs="0" maxOccurs="unbounded"/>".

The present document specifies the XML AIT schema in ETSI TS 102 796 [] for the "version” element which is changed as follows. This line:

```
<xsd:element name="version" type="ipi:Version"/>
```

is changed to:

```
<xsd:element name="version" type="mhp:unsignedInt31Bit"/>
```

And this type definition is added to the end of the schema, just before the "</xsd:schema>":

```
<xsd:simpleType name="unsignedInt31Bit">
    <xsd:restriction base="xsd:unsignedInt">
        <xsd:maxInclusive value="2147483647"/>
    </xsd:restriction>
</xsd:simpleType>
```

NOTE 1: These changes mean that the version field is encoded in the XML file in decimal, instead of hex. They also increase the range, so it is an unsigned 31-bit integer instead of an unsigned 8-bit integer. This avoids wrapping of version numbers in operator applications that are likely to be long-lived and have far more than 256 revisions.

NOTE 2: The requirement in TS 102 034 [28] that ipi:Version fields are modulo 256 does not apply to this field, because the type is changed so it is not ipi:Version.

The terminal shall initially trust the XML AIT file found through the discovery process for the purposes of:

- identifying operators where a bilateral agreement is in place from the list of supported operators (using at least the `orgId`); and
- finding the location of the encrypted application package. The terminal shall use at least the `orgId` within the XML AIT to identify operators where a bilateral agreement is in place (see clause 6.6.1).

The terminal shall ignore the XML AIT if there is no bilateral agreement with the operator in place.

NOTE: An attacker modifying the signalling to point at a different XML AIT will not be able to trick terminals into installing a malicious operator application because of the requirement in clause 11.3.4.5 for the terminal to authenticate the package before extracting the ZIP file.

Terminals supporting the present document in combination with TS 102 796 V1.7.1 or later shall additionally support the requirements relating to the GraphicsConstraints element and may support graphics co-ordinate systems greater than 1280x720 for operator applications as long as the terminal can composite operator applications and regular HbbTV applications with different coordinate spaces. Terminals supporting the present document in combination with versions of TS 102 796 before V1.7.1 shall silently ignore the presence of a GraphicsConstraints element.

6.1.5.2 AIT Acquisition

The terminal shall tune to the DVB services identified by the DVB URIs discovered in clause 6.1.2 and search for an AIT table with the application type 0x10 according to the rules specified in clause 7.2.3 of ETSI TS 102 796 [].

The terminal shall look for AIT entries with application usage types supported by the terminal. The broadcast AIT shall conform to the profile in clause 7.2.3.1 of ETSI TS 102 796 [] with the additional requirements in table 8. Additionally, the application version descriptor defined in clause 7.2.2 is mandatory. The terminal should ignore AIT entries that do not meet these requirements for the purpose of operator application discovery.

Table 8: AIT Profile

Clause	Page	Status	Notes
5.2.4 Application control codes	16	M	The following control codes shall be supported: `0x02 PRESENT`
5.2.10 Application usage	22	M	Usage type 0x80 or 0x81 shall be supported: 0x80 privileged operator application 0x81 operator specific operator application
5.3.5.5 Application usage descriptor	37	M	See application usage above.
5.3.5.6 User information descriptors	38	M	
5.3.5.6.1 Application name descriptor	38	M	This may be used to populate an operator application's entry in the list of supported operators (see clause 6.6.1).
5.3.5.6.2 Application icons descriptor	38	M	This may be used to populate an operator application's entry in the list of supported operators (see clause 6.6.1).
5.3.6 Transport protocol descriptors	40	M	The following `protocol_ids` shall be supported: 0x0001 object carousel over broadcast channel
5.3.7 Simple application location descriptor	43	M	It shall point to the location of the encrypted application package, see clause 11.3 regarding the format of the encrypted application package. The initial_path_bytes shall have the value of the location of the encrypted application package relative to the top level directory of the object carousel signalled in the transport_protocol_descriptor. EXAMPLE: "`application/opapp.pkg` "
NOTE: The clause numbers and page numbers in this table refer to ETSI TS 102 809 [24].			

The terminal shall initially trust the AIT table found through the discovery process for the purposes of:

- identifying operators where a bilateral agreement is in place, i.e. from the list of supported operators (see clause 6.6.1). (using at least the `orgId`); and
- finding the location of the encrypted application package.

NOTE: An attacker modifying the signalling to point at a different XML AIT will not be able to trick terminals into installing a malicious operator application because of the requirement in clause 11.3.4.5 for the terminal to authenticate the package before extracting the ZIP file.

The terminal shall ignore the AIT if any of the following apply:

- there is no matching bilateral agreement in place;
- any transport_protocol_label referenced from the application_descriptor references a transport protocol descriptor that is conveyed in the common descriptors loop;
- there exists a second application loop entry in the broadcast AIT used to discover the operator application that has the same organisation_id and application_id;
- No application version descriptor is included.

Terminals supporting the present document in combination with TS 102 796 V1.7.1 or later shall additionally support the requirements relating to the graphics_constraints_descriptor and may support graphics co-ordinate systems greater than 1280x720 for operator applications as long as the terminal can composite operator applications and regular HbbTV applications with different coordinate spaces. Terminals supporting the present document in combination with versions of TS 102 796 before V1.7.1 shall silently ignore the presence of a graphics_constraints_descriptor.

6.1.6 Deciding which operator applications to install

The result of the previous processes is a number of (XML) AITs each corresponding to an operator application where a bilateral agreement is in place for the terminal. Which of these are installed depends on a number of factors:

- For operator-specific operator applications, the appropriate operator-specific operator application may be pre-installed or be automatically installed after discovery.
- The bilateral agreement itself may include provision for automatic installation of an operator application discovered when the user inserts a corresponding CICAM.
- The terminal may offer the user an explicit choice through some kind of menu of operator applications.
- The terminal may offer the user an implicit choice by adding operator applications to a menu of operators that also determines the algorithm for channel list and channel number setup.
- Terminals may be limited to installing a finite number of operator applications which could be one (e.g. for a white label set-top box intended to support a single operator-specific operator application).

For those operator applications that are to be installed, the terminal shall download and authenticate the encrypted application package, as specified in clauses 6.1.7 and 11.3.

6.1.7 Encrypted application package download

6.1.7.1 Introduction (informative)

The present document defines three mechanisms by which an encrypted application package can be downloaded:

- HTTP over TLS via broadband IP;
- DSM-CC object carousel over broadcast, and
- CIPlus Auxiliary File System.

Each of these is described in more detail below.

6.1.7.2 Encrypted application package download via IP

The terminal shall perform an HTTP GET request to download the encrypted application package from the URI defined in the location descriptor of the retrieved XML AIT (see clause 6.1.5.1). The terminal shall provide the HTTP User-Agent as defined in clause 7.3.2.4 of ETSI TS 102 796 [] when requesting the encrypted application package. The terminal shall observe any TLS mutual authentication requests as specified in clause 11.2.1. The terminal shall reject any encrypted application package where the Content-Type header does not have the value `application/vnd.hbbtv.opapp.pkg`. The terminal shall decrypt and verify the operator application package as defined in clause 6.1.8.

The terminal may experience connection or retrieval issues when downloading the encrypted application package. A terminal shall attempt to retry the download request for a maximum of 3 attempts (i.e. the initial request and two retry attempts) with a random value between 60 and 600 seconds between the requests. If the download process fails after all the attempts, then the terminal shall follow the process defined in clause 6.1.9.

6.1.7.3 Encrypted application package download via DSM-CC object carousel

The terminal shall use the simple application location descriptor in the retrieved AIT (see clause 6.1.5.2) to download the encrypted application package onto the terminal. The terminal shall decrypt and verify the encrypted application package as defined in clause 6.1.8.

If the DSM-CC object cannot be accessed (e.g. the object is not present in the carousel or if the carousel cannot be mounted due to another request) , the terminal should attempt to retry the download. If the download continues to fail, the terminal shall follow the process defined in clause 6.1.9.

6.1.7.4 Encrypted application package download from CI Plus AuxFS

When the applicationTransport field in the XML AIT is set to CITransportType, the Terminal shall use the applicationLocation element as the reference of the encrypted application package to be obtained from the CI Plus Auxiliary File System. The terminal shall decrypt and verify the encrypted application package as defined in clause 6.1.8.

A Terminal that has installed an OpApp from the CI Plus Auxiliary File System shall allow such OpApp access to files from the CI Plus Auxiliary File System by use of the URL format as defined in clause 9.2 of HbbTV Specification.

6.1.8 Decrypt, verify, unpack and installation of the application package

The terminal shall decrypt the encrypted application package as defined in clause 11.3.4.4 using the Terminal Packaging Certificate and corresponding private key. The terminal shall verify the signature of the decrypted application ZIP package as specified in clause 11.3.4.5.

The terminal shall consider the application package as valid and verified if all of the following are true:

- The application zip package passed the verification process defined in clause 11.3.4.5.
- For application packages signalled by a broadcast AIT, the application loop entry from the initially trusted broadcast AIT matches the `opapp.ait` file contained inside the package.
- For application packages signalled by an XML AIT, the initially trusted XML AIT file matches the `opapp.aitx` from inside the package.
- When an already installed operator application is being updated, if a minimum application version number was provided when the package was last updated (or installed if this is the first update) then:
    - the version number in the application package to be installed is greater than or equal that minimum version number;

  otherwise if no minimum application version number was provided at that time;

    - the version number in the application package to be installed is higher than the version number of the currently installed operator application.
- The combined uncompressed and extracted size of the operator application files is smaller than the maximum permitted, subject to the bilateral agreement.

Once verified, the terminal shall copy the operator application files into the terminal's persistent storage area. If this is an update of a previously installed operator application, all of the previously stored application files shall be deleted. Terminals may either stop the operator application while the installation is happening or alternatively keep the operator application running until the installation has completed and then restart it. In the latter case, the operator application shall not see any files from the update until it is restarted and the deletion of the previously stored application files shall only happen after the operator application has been restarted.

6.1.9 Installation failures

6.1.9.1 Installation failure overview

During the discovery and installation of an operator application, there are several points in the process where failures may occur. These failure conditions can be divided into two categories:

a) If the installation request is initiated for the first time and there is no previous operator application from this operator installed, the terminal shall manage the user messaging as defined in clause 6.1.9.2.

b) If the installation/update is initiated by an existing operator application, the operator application shall manage the user messaging as defined in clause 6.1.9.3.

6.1.9.2 Failure handling on first-time installation

Terminals may encounter failure conditions during the first-time installation of an operator application. This clause outlines how a terminal handles the failure.

Terminals may have initiated the first-time installation of the operator application as described by clause 5.2.1. If a failure occurs during a user-initiated installation process, the user shall be able to determine that the installation has failed. The terminal may provide additional information to the user and/or error codes that may be useful in diagnosing the cause of the failure.

The terminal may offer the user the option to attempt the installation again immediately or at a later time. Alternatively a terminal UI may be subject to the bilateral agreement.

6.1.9.3 Failure handling when updating an operator application

Terminals may encounter installation failures during an update requested by the operator application using the `opAppRequestUpdate()` method defined in clause A.2.2.2. The method allows for the action to take place immediately or at a time convenient to the user.

If the operator application update was requested to occur immediately, then on detection of the update failure;

- If the operator application is no longer running then the terminal shall immediately launch the existing version of the operator application with the appropriate context as defined in clause 7.1.1 and the status launch parameter as defined in clause 7.1.2.
- If the operator application is still running then it can detect the failure via `onOpAppUpdate` or an `OpAppUpdate` event as defined in clause A.2.2.1.

If the operator application update was requested to occur at a convenient time, the terminal shall launch the existing version of the operator application at the next appropriate time.

Regardless of when the update was requested to occur, if the operator application is no longer running then when the terminal re-launches the application, the terminal shall provide the launch and startup context signalling as defined in clause 7.1.1 with the appropriate values and include the status launch parameter defined in clause 7.1.2 with a value of `updateFailed`.

The terminal shall not attempt the update process again unless requested by the operator application.

6.2 Updating operator applications

Once an operator application has been installed, it is solely responsible for updates, the present document does not define any mechanism or responsibility for the terminal to initiate the update of an operator application.

Operator applications are able to request they be updated by calling the `opAppRequestUpdate` method (see clause A.2.2.2). The operator application may request the update happen immediately or be delayed to happen at a time convenient for the user (e.g. when the terminal is in standby for a significant period). The operator application may add additional parameters required in the request. The terminal shall append the parameters to the result from the discovery process.

When the terminal executes an update request from an operator application, the terminal shall run all discovery methods defined in the list of supported operators for that operator application in order to find an operator application with the same organisation_id and application_id and a different version number.

If the terminal uses multiple discovery methods to find updates for an operator application, and at least one of those discovery methods succeeds in finding an operator application with the same organisation_id and application_id, then any failures of other discovery methods shall be ignored.

The operator application update mechanism may be used when an operator has a CICAM delivered version of an application and a broadband delivered version of application and an installed version of the CICAM delivered application detects that broadband has now been connected. In order to achieve this;

- The broadband-delivered version of the application would need to have a higher version number than the CICAM-delivered version and

- The bilateral agreement would need to give the broadband-delivered version of the application precedence over the CICAM delivered application when both versions are discovered and available. This would apply both on initial installation of the application and on update.

The final step in a update of an operator application is the reboot of the operator application using the newly updated version with "status=updateSuccessful" as defined in clause 7.1.2. If the operator application fails to boot correctly (e.g. fatal errors, running out of memory, exiting, ...) then the terminal should keep running the version from before the update.

6.3 Operator application lifecycle

6.3.1 Introduction

The operator application lifecycle is determined by the following four factors:

1) The application state model.

2) The activation of the operator application by the user or by some other means.

3) The termination of the operator application under certain conditions.

4) The terminal entering and leaving modes where linear TV is active.

5) The bilateral agreement between the terminal manufacturer and the operator.

The operator application lifecycle is independent of that of any regular HbbTV® application or any broadcast application signalling.

6.3.2 Starting and stopping operator applications

6.3.2.1 Summary (Informative)

A number of ways are defined by which an operator application can be started:

- Directly by the end-user (e.g. by using a 'select source' menu to switch to the operator application 'source' or by pressing an operator application key (see 10.1.3) like EPG or channel list or by pressing a dedicated operator key). A 'select source' menu may use the application name and icon from the (XML) AIT discovered when the operator application was installed (see clause 6.1.5).
- By an already running operator application (via the `createApplication()` method).
- By the terminal (e.g. after a power cycle in cases where the operator application was the active 'source' before the power cycle).

See also clause 5.2.1 for more information.

A number of ways are defined by which an operator application may be stopped:

- By the operator application itself calling the `Application.destroyApplication` method.
- By the terminal, under error conditions (e.g. out of memory).
- By the terminal, for reasons specified in the bilateral agreement.
- Directly by the end-user (e.g. by using a 'select source' menu to switch to a source other than the operator application one).

NOTE: An operator application is not stopped by the end-user using the EXIT key or equivalent.

6.3.2.2 Starting operator applications

The initial document of an operator application shall be "index.html" in the top level directory of the application ZIP file. When starting an installed operator application, the terminal shall load and run this "index.html" file. The "index.html" shall be run using a URL constructed according to clause 9.4.2 with the appropriate launch and startup contexts defined in clause 7.1.1 so that the operator application can present the appropriate starting page.

NOTE 1: The bilateral agreement may specify that certain keys on the remote control trigger particular entry points (e.g. EPG, channel list). For many of these entry points, the operator application will need to call `opAppRequestForeground` in the handler for the load event of its initial document in order to present the requested user interface.

NOTE 2: The bilateral agreement needs to define the power-on behaviour, specifically if the operator application is always started when the terminal is powered-on or if starting the operator application depends on some combination of what the terminal was doing before the power was turned off and what the terminal will be doing immediately after it is turned on.

6.3.2.3 Stopping operator applications

If an operator application is stopped by the terminal under error conditions (e.g. out of memory) then the operator application shall be restarted.

If an operator application is stopped for either of the following reasons then the behaviour will be determined by the bilateral agreement:

- By the operator application itself calling the `Application.destroyApplication` method.
- By the terminal, for reasons specified in the bilateral agreement.

EXAMPLE: If the bilateral agreement requires an operator application to be stopped when the user chooses a channel outside the operator's channel list then it would not be appropriate to return to a home page or source selection UI from which the user could choose to start the operator application again.

When the terminal enters a mode where linear TV is no longer active (e.g. an HDMI input or an application from a global VOD service provider is selected), the operator application may be stopped or it may continue to run in order to optimise performance when the terminal subsequently leaves such a mode. See clause 6.3.3.1.

The bilateral agreement may define some conditions under which an operator application is required to be restarted without the terminal returning to a home page or source selection UI. The terminal shall provide the restart value of the `<launch location>` as defined in clause 7.1.1 and set the status parameter to error as defined in clause 7.1.2. Great care should be taken to avoid the user becoming trapped by an application error.

6.3.2.4 Co-existence of multiple operator applications

Two operator applications cannot run at the same time. One operator application can launch another one. In that case the first operator application is terminated.

Terminals may support simultaneous installation of more than one operator application including but not limited to the following scenarios;

- Operator applications with the same physical layer (e.g. multiple satellite operator applications) or
- Operator applications with different physical layers (e.g. one terrestrial and one satellite) or
- One or more operator applications linked to a broadcast physical layer (e.g. satellite) and one or more operator applications not linked to any broadcast physical layer at all (e.g. only broadband delivered content).

See clause 9.2 for more information on the consequences of these options for channel lists.

6.3.3 Operator application states

6.3.3.1 Introduction

A running operator application shall be in one of five states: background state, foreground state, transient state, overlaid foreground state or overlaid transient state. These states are listed for information in table 9.

Table 9: Operator application states (informative)

State	Example	Focus	opAppState property (See clause A.2.2.1)	Regular app visibility
Background	Operator application is not presenting any UI and is monitoring for events such as a channel change	No	background	Yes
Foreground	Operator application is presenting an EPG	Yes	foreground	Optional (see note)
Transient	Operator application is presenting a zapping banner	Yes	transient	Yes
Overlaid foreground	Terminal UI appears on top of the operator application EPG	No	overlaid-for eground	Optional (see note)
Overlaid transient	Terminal UI appears on top of operator application zapping banner	No	overlaid-transient	Yes
NOTE: On terminals that support the optional behaviour "Running regular HbbTV® applications with an operator application in the foreground" as defined in clause 6.5.3, a regular HbbTV® application may be visible under certain circumstances. On terminals not supporting that behaviour, regular HbbTV® applications are required to be stopped when the operator application is in this state and hence cannot be visible.				

When launched by the terminal, operator applications shall be in the background state.

If an operator application successfully launches a second operator application using `createApplication`, the first operator application shall be killed and the second operator application shall inherit the operator application state and the value of the countdown timer (see below), when applicable, from the first operator application.

The state of the operator application is independent of any regular HbbTV® application that may be running, except that the operator application and the regular HbbTV® application shall not simultaneously have focus (where focus is defined in ETSI TS 102 796 []).

If the user triggers the "EXIT or comparable button" mechanism as defined in clause 10.2.2.1 of ETSI TS 102 796 [], this shall have no effect on an operator application state except when (a) the operator application is in the foreground or transient state and (b) it has not registered to receive key events for the EXIT or comparable button. In this case a transition to the background state is forced (see clause 6.3.3.3).

If the terminal kills the operator application due to an error condition (e.g. out of memory), the operator application shall be restarted. The terminal shall provide the restart value of the `<launch location>` as defined in clause 7.1.1 and set the status parameter to error as defined in clause 7.1.2. If the operator application calls the `destroyApplication()` method, it shall not be automatically restarted and hence the user shall return to the UI provided by the terminal.

The behaviour of all video/broadcast objects in the operator application shall be as defined in clause 9.1.

Interactions between the operator application lifecycle and the lifecycle of regular HbbTV® applications are defined in clause 6.5.2.

Terminals may enter modes where linear TV is no longer active (e.g. HDMI input being selected or an application from a global VOD service provider being selected). When this happens, terminals shall either;

- Stop running an operator application on entry to such modes (see clause 6.3.2.3) and re-start it on return to linear TV; or

- Transition the operator application to the background state and 'opapp-silent" context on entry and, on exit from that mode, transition the operator application to an appropriate context (and perhaps a different state) based on how the exit was triggered.

The bilateral agreement may address this.

6.3.3.2 Foreground state

The terminal shall move the operator application to this state when any of the following are true:

- The operator application is in the background or transient state and it makes a successful call to the `opAppRequestForeground` method:
  - The call shall be successful if it was made within a handler for keydown, keyup or keypress events in the operator application.
  - The call shall be successful if it was made within a handler for a click event for a notification requested by the operator application and activated by the user.
  - The call shall be successful if it was made within a handler for the load event of the initial document of the operator application only when the operator application is started.
  - The call shall be successful if it was made within a handler for `OperatorApplicationContextChange` events in the operator application.
  - The call shall be successful if it was made within the handler for onPlayStateChange events due to a transient error (transition from connecting to connecting) or permanent error (transition from either connecting or presenting to unrealized).
  - The call shall be successful if it was made within the handler for an ApplicationUnloaded event due to the termination of a regular HbbTV application launched by the OpApp as defined in table A.1 for the application/oipfApplicationManager embedded object
  - The call shall be successful if it was made within the handler for an onMessage event of the WebSocket interface as a result of a message received from a regular HbbTV application using the communication mechanism defined in clause 15.
  - The call shall fail if none of the above conditions apply.
- The operator application is in the overlaid foreground state and the terminal removes the UI which was overlaying the operator application.
- While interacting with the terminal user interface, the user selects a UI element that has been replaced by the operator application (see clause 6.4), no simulated key event is generated (see clause 10.1.3) and the bilateral agreement specifies that the foreground state is to be used when that UI element is selected.

NOTE 1: If the user selects UI elements from the terminal UI such as guide or info, the terminal may send a simulated key event to the operator application and allowing the operator application to request the appropriate state change.

The following shall apply when an operator application is running in foreground state:

- The operator application has full access to a graphic plane (see clause 9.3).

NOTE 2: Calls to `bindToCurrentChannel()` can be made by the operator application as soon as it is moved to the foreground state. This will allow the video/broadcast object in the operator application to present the current channel subject to the restrictions described in clause 9.1.4.

- The operator application has access to the operator application keys, as defined by clause 10.1.3.
- The operator application has access to the regular HbbTV applications keys as defined by clause 10.1.3.
- The operator application is able to request the terminal to move it to the background state or the transient state.

6.3.3.3 Background state

The terminal shall move the operator application to this state when any of the following are true:

- The operator application is in any other state and it makes a successful call to the `opAppRequestBackground` method.
- The operator application is in the transient or overlaid transient state and the countdown timer set by the terminal reaches zero.
- The operator application is in the foreground state or in the transient state, it has not registered to receive key events for the EXIT or comparable button and the user triggers the "EXIT or comparable button" mechanism as defined in clause 10.2.2.1 of ETSI TS 102 796 [].
- The operator application is in any state other than background and successfully starts a broadcast-independent regular HbbTV® application using the version of the `createApplication` method defined in clause A.2.2.2 .
- The terminal enters a mode where linear TV is no longer active (e.g. an HDMI input or an application from a global VOD service provider are selected) and the operator application is not killed.

On entry to this state, any broadcast channel being presented by the operator application in a video/broadcast object shall be presented under the control of the terminal instead (the scaling applied by the operator application shall be ignored by the terminal). If there was no such channel, the terminal UI (graphics and/or video) is not defined by the present document.

The following shall apply when an operator application is running in background state:

- The operator application has no access to any graphic plane and is not visible to the user.
- If the operator application is not in opapp-silent context;
  - The operator application has access to the operator application keys, as defined by clause 10.1.3.
  - The operator application might have access to some of the regular HbbTV application keys and operator application keys as defined by clause 10.1.3.
  - The operator application is able to request the terminal to move it to the foreground state or the transient state.
  - Any video/broadcast object held by the operator application is in the unrealized state as defined in clause A.2.4.1.
  - The operator application shall not be able to use scarce media resources except as defined in clause 9.1.3.
- Otherwise if the operator application is in opapp-silent context;
  - The operator application does not receive any key events.
  - The terminal may ignore requests by the operator application to show notifications (to be defined in the bilateral agreement).
  - Any video/broadcast object and/or broadcast supervisor held by the operator application is in the unrealized state as defined in clause A.2.5.
  - The operator application is still able to receive events due to web features such as timers or `XMLHttpRequest`s.
  - If the user initiates a return to linear TV, this shall cause the terminal to send a `OperatorApplicationContextChange` event to the operator application indicating a new context for the operator application. Based on the specified context, the operator application can then call either `opAppRequestForeground` or `opAppRequestTransient` as appropriate.

NOTE: A consequence of the above is that, of all the conditions specified in clause 6.3.3.2 or 6.3.3.4 for a call to either `opAppRequestForeground` or `opAppRequestTransient` to succeed, the only one that can apply in opapp-silent context is when the method is called in a handler for the `OperatorApplicationContextChange` event. Hence an operator application is unable to exit the opapp-silent context on its own initative. The user must initiate a return to linear TV, causing an `OperatorApplicationContextChange` event.

- The operator application shall not be able to use scarce media resources at all.
-

6.3.3.4 Transient state

The terminal shall move the operator application to this state when any of the following are true:

- The operator application is in the foreground or background state and it makes a successful call to the `opAppRequestTransient` method:
  - The call shall be successful if it was made within a handler for ChannelChangeSucceeded events in the operator application where the `quiet` argument to the ChannelChangeSucceeded event was not '1' or '2'.
  - The call shall be successful if it was made within a handler for ChannelChangeError events in the operator application.
  - The call shall be successful if it was made within a handler for keydown, keyup or keypress events in the operator application.
  - The call shall be successful if it was made within a handler for click events for a notification requested by the operator application and activated by the user.
  - The call shall be successful if it was made within a handler for `OperatorApplicationContextChange` events in the operator application.
  - The call shall be successful if it was made within a handler for the load event of the initial document of the operator application only when the operator application is started.
  - The call shall be successful if it was made within the handler for PlayStateChange events in the operator application where the state parameter is 0 (“unrealized”) or 1 (“connecting”).
  - The call shall be successful if it was made within a handler for `onProgrammesChanged` events in the operator application.
  - The call shall be successful if it was made within the handler for an ApplicationUnloaded event due to the termination of a regular HbbTV application launched by the OpApp as defined in table A.1 for the application/oipfApplicationManager embedded object
  - The call shall be successful if it was made within the handler for an onMessage event of the WebSocket interface as a result of a message received from a regular HbbTV application using the communication mechanism defined in clause 15.
  - The call shall fail if none of the above conditions apply.
- The operator application is in the overlaid transient state and the terminal removes the UI which was overlaying the operator application.
- While interacting with the terminal user interface, the user selects a UI element that has been replaced by the operator application (see clause 6.4), no simulated key event is generated (see clause 10.1.3) and the bilateral agreement specifies that the transient state is to be used when that UI element is selected.

NOTE 1: If the user selects UI elements from the terminal UI such as guide or info, the terminal may send a simulated key event to the operator application and allowing the operator application to request the appropriate state change.

On entry to this state from the foreground, overlaid foreground or background state, the terminal shall start a countdown timer of 60 seconds. If the operator application is already in the transient state and makes a call to the `opAppRequestTransient` method and one of the requirements for the call to be successful listed above is met, then the call shall be successful and the 60 second countdown timer shall be restarted from the beginning.

On entry to this state, any broadcast channel being presented by the operator application in a video/broadcast object shall be presented under the control of the terminal instead (the scaling applied by the operator application shall be ignored by the terminal). If there was no such channel, the terminal UI (graphics and/or video) is not defined by the present document.

NOTE 2: If an operator application transitions from the foreground state to this state and a regular HbbTV® broadcast-related application is started then video will be presented under the control of the terminal until that regular HbbTV® application successfully calls `bindToCurrentChannel`.

The following shall apply when an operator application is running in transient state:

- The operator application has full access to a graphic plane (see clause 9.3).
- The operator application has access to the operator application keys, as defined by clause 10.1.3.
- The operator application has access to the regular HbbTV applications keys as defined by clause 10.1.3.
- The operator application is able to request the terminal to move it to the foreground state or background state.

When an operator application in the transient state gains focus, any running regular HbbTV® application shall lose focus and become "blurred". The terminal may wish to indicate this change to the user, perhaps by making any UI from the regular HbbTV® application darkened or faded to a certain degree. Whatever approach is taken by the terminal, any UI objects presented by the regular HbbTV® application shall remain visible to the user (although not necessarily legible), assuming that they are not obscured by any UI from the operator application.

6.3.3.5 Overlaid foreground state

The terminal shall move the operator application to this state when any of the following are true:

- The terminal is displaying some UI and the operator application is in the background or overlaid transient state and it makes a successful call to the `opAppRequestForeground` method:
  - The call shall be successful if and only if one or more of the conditions for a successful entry to the foreground state apply, as defined in clause 6.3.3.2.
- The operator application is in the foreground state and the terminal starts to display some UI on top of the operator application.

The following shall apply when an operator application is running in overlaid foreground state:

- The operator application has full access to a graphic plane (see clause 9.3), although some or all of it may be overlaid by some terminal UI.
- The operator application has access to the operator application keys, as defined by clause 10.1.3.
- The operator application may have access to the regular HbbTV applications keys and operator application keys as defined by clause 10.1.3.
- The operator application is able to request the terminal to move it to the background state.

6.3.3.6 Overlaid transient state

The terminal shall move the operator application to this state when any of the following are true:

- The terminal is displaying some UI and the operator application is in the overlaid foreground or background state and it makes a successful call to the `opAppRequestTransient` method:
  - The call shall be successful if and only if one or more of the conditions for a successful entry to the transient state apply, as defined in clause 6.3.3.4.

- The operator application is in the transient state and the terminal starts to display some UI on top of the operator application.

On entry to this state from the foreground, overlaid foreground or background state, the terminal shall start a countdown timer of 60 seconds. If the operator application is already in the overlaid transient state and makes a call to the `opAppRequestTransient` method and one of the requirements for the call to be successful listed above is met, then the call shall be successful and the 60 seconds countdown timer shall be restarted from the beginning.

On entry to this state, any broadcast channel being presented by the operator application in a video/broadcast object shall be presented under the control of the terminal instead (the scaling applied by the operator application shall be ignored by the terminal). If there was no such channel, the terminal UI (graphics and/or video) is not defined by the present document.

NOTE: If an operator application transitions from the foreground state to this state and a regular HbbTV® broadcast-related application is started then video will be presented under the control of the terminal until that regular HbbTV® application successfully calls `bindToCurrentChannel`.

The following shall apply when an operator application is running in overlaid transient state:

- The operator application has full access to a graphic plane (see clause 9.3), although some or all of it may be overlaid by some terminal UI.
- The operator application has access to the operator application keys, as defined by clause 10.1.3.
- The operator application may have access to the regular HbbTV applications keys and operator application keys as defined by clause 10.1.3.
- The operator application is able to request the terminal to move it to the foreground state or background state.

On a transition from transient to overlaid transient, the requirements of clause 6.3.3.4 concerning "any UI objects presented by the regular HbbTV® application" shall apply.

6.3.4 Standby states

Privileged and operator-specific applications shall not run when the terminal is entering in PASSIVE_STANDBY as defined in clause 7.3.3.1 of [2] or PASSIVE_STANDBY_HIBERNATE as defined in clause 7.3.3.1 of [2] states. When entering the passive states, the terminal shall stop executing any running operator applications.

Prior to entering the passive states, the terminal may enter ACTIVE_STANDBY and terminals may continue to run the operator applications subject to the bilaterial agreement for a short period of time. The terminal shall notify the operator application by raising an `onPowerStateChange()` indicating the new ACTIVE_STANDBY state. The `powerState` property shall indicate the ACTIVE_STANDBY state and the `timeCurrentPowerState` (defined in clause 7.3.3 of [2]) shall be set accordingly. The bilaterial agreement shall state the period of time that an operator application has during this state before the terminal moves into either one of PASSIVE_STANDBY or PASSIVE_STANDBY_HIBERNATE states and terminates the running operator application. The availability of resources such as tuners and network access in ACTIVE_STANDBY state may be defined in the bilateral agreement .

When exiting passive standby states, the terminal is not required to automatically re-launch any operator applications just because they were running when the TV entered the standby states, although operator applications may be launched for other reasons. E.g. if the bilateral agreement says that a particular operator application is always launched when exiting standby, or if the terminal automatically resumed displaying a source that is provided by an operator application.

NOTE: The present document does not define a way for operator applications to run and manage different power states.

6.4 UI elements provided by an operator application

Table A.2 identifies some groups of UI elements that may be suppressed by the terminal in order to be replaced by an operator application. Terminals shall support suppression of those groups of UI elements marked as "M" in that table.

Subject to the bilateral agreement, terminals may support suppression of other groups of UI elements from that table and further UI elements defined by the manufacturer and operator (using values from the user defined range).

The operator application shall be able to define which UI elements it will provide instead of the terminal by using the `replaceUIElements` method of the Configuration class as defined in clause A.2.1.3.

Once the operator application has called that method and as long as the operator application is running, the terminal shall suppress all UI elements and related functionality as defined by that method within the limits set by the bilateral agreement (and as known to the terminal in the supported operators list described in clause 6.6.1). When an operator application terminates, UI elements and related functionality that it was providing shall revert to being provided by the terminal until an operator application is started and calls the `replaceUIElements` method.

When the user selects a UI element that has been replaced by the operator application, the following shall apply:

- If the operator application is running in the background state and the user selection corresponds to an operator application reserved key (see clause 10.1.3) that the operator application has in its keyset then a simulated key event shall be generated to the operator application.
- If the operator application is running in the background state and the user selection either does not correspond to an operator application reserved key (see clause 10.1.3) or corresponds to an operator application reserved key that the operator application does not have in its keyset then the operator application shall be brought into the foreground or transient state as defined in clause 6.3.3.

NOTE: The state that the operator application transitions to will depend on the user selection and on the bilateral agreement. Foreground will be appropriate for some UI elements and transient to others.

The bilateral agreement may define UI elements that, when selected by the user, cause an operator application to be started if it is not already running. The reason for the operator application being started shall be encoded using the startup and launch context as defined in clause 7.1.1.

6.5 Regular HbbTV® application signalling and lifecycle

6.5.1 Application signalling

Terminals shall monitor and act on the HbbTV® application signalling, as defined in clause 7.2.3.1 of ETSI TS 102 796 [], depending on the state of the operator application as follows:

- Foreground state and overlaid foreground state: terminals shall detect the presence of an application overlay descriptor (see clause 7.2.1) in the "common" (outer) loop of the AIT and act on it as defined in clause 9.1.4 for blocking video presentation. If a terminal runs regular HbbTV® applications when an operator application is in the foreground state or overlaid foreground state then in addition the requirements in clause 6.5.3 shall be followed.
- Transient state and overlaid transient state: terminals shall act on this signalling as defined in ETSI TS 102 796 [].
- Background state: terminals shall act on this signalling as defined in ETSI TS 102 796 [].

6.5.2 Starting and stopping regular HbbTV® applications

The following requirements are in addition to those defined in clause 6.2 of ETSI TS 102 796 []:

- Operator applications shall be able to launch regular HbbTV® broadcast-independent applications as defined in clause A.2.2.2 of the present document. As defined in clause 6.2.2.6.1 of ETSI TS 102 796 [], any current broadcast service shall cease to be selected - logically equivalent to selecting a "null service". If the operator application is in the foreground or overlaid foreground state, then it shall move to the background state if the regular HbbTV® application is successfully launched.
- An operator application transitioning from foreground or overlaid foreground state to one of the transient, overlaid transient or background states when a broadcast channel is being presented shall result in the HbbTV® application lifecycle being obeyed which may require the terminal to start a broadcast-related application.

NOTE: An operator application may transition from foreground to transient in order to show a channel banner. Until the channel banner times out, this would result in both the operator application and any broadcast-related HbbTV® application on that channel being visible at the same time.

- When an operator application in the background, transient or overlaid transient state changes the broadcast channel using the BroadcastSupervisor class, the terminal shall follow the HbbTV® application lifecycle from clause 6.2 of ETSI TS 102 796 [] which may require the terminal to stop or start a broadcast-related application.
- When the terminal successfully loads a regular HbbTV application associated with a given channel, the terminal shall trigger an `ApplicationLoaded` event on the `BroadcastSupervisor` object. If the regular HbbTV application fails to successfully load, the terminal shall trigger an `ApplicationLoadError` on the `BroadcastSupervisor` object, as defined in clause 6.2.2.5.6 of ETSI TS 102 796 [1] and clause 7.2.1.2 of OIPF DAE [2].

  Any listeners attached to the `ApplicationLoaded` and `ApplicationLoadError` events shall receive an object of type Application class. The application object shall only contain the `privateData.currentChannel` property with a value corresponding to the channel associated with the regular HbbTV application. All other properties and methods defined on the application class shall return undefined.
- If an operator application successfully calls the `opAppRequestForeground` method (according to the requirements defined in clause 6.3.3.2), the terminal should kill any running regular HbbTV® application otherwise the requirements in clause 6.5.3 shall be followed.

6.5.3 Running regular HbbTV® applications with an operator application in the foreground

If the terminal supports running regular HbbTV® applications at the same time as an operator application in the foreground or overlaid foreground state, the following shall apply:

- Any running HbbTV® application shall continue to run when the operator application transitions to the foreground or overlaid foreground state:
  - The broadcast application signalling in the AIT shall be obeyed (as defined in clause 6.2.2 of ETSI TS 102 796 []).
  - The regular HbbTV® application shall not have focus.
  - The regular HbbTV® application shall be hidden if video presentation is blocked (as defined by clause 9.1.4) or alignment between graphics and video is changed (e.g. due to video scaling by an operator application).

If the regular HbbTV® application is killed by the terminal due to resource conflicts as defined in clause 9.1.2, the broadcast application signalling in the AIT shall be ignored until the operator application enters the transient, overlaid transient or background state.

If the regular HbbTV application is killed by the terminal for any other reason (e.g. destroying itself, AIT removal/update), the broadcast application signalling in the AIT may be ignored until the operator application enters the transient, overlaid transient or background state.

6.5.4 Application signalling for operator application created channels

When an operator application created channel is selected, terminals shall act upon any application signalling that was included in the metadata for that channel when the Channel object was created. Such signalling shall be processed as specified for a DVB-I service instance delivered by MPEG DASH in ETSI TS 102 796 [] clause O.3 and O.4. This shall apply whatever the state of the operator application.

If the operator application is in the foreground state or overlaid foreground state and the terminal runs regular HbbTV applications when an operator application is in those states then in addition the requirements in clause 6.5.3 shall be followed.

6.6 Multiple operator applications

6.6.1 Supported operators

Terminals shall include a list of those operators where a bilateral agreement is in place (see annex D). This list shall be populated with at least the following information necessary for the installation of operator applications:

- Which of the discovery methods in clause 6.1.2 are applicable to the operator including any additional information necessary for the applicable method(s) as follows:
  - Which broadcast NIT/BAT to use for either of the following discovery methods:
    - Broadcast NIT/BAT with URI_linkage_descriptor with operator FQDN (clause 6.1.3.2).
    - Broadcast NIT/BAT with URI_linkage_descriptor with URI of AIT (clause 6.1.3.3).
  - An FQDN for the method "Hardwired in terminal with operator FQDN" (clause 6.1.3.5).
  - A URL for the method "Hardwired in terminal with URI of XML AIT" (clause 6.1.3.6).
- The organisation_id of the operator (as used in clauses 6.1.5.1 and 6.1.5.2):
  - Also a definition of any other fields in the (XML) AIT needed to identify if a bilateral agreement is in place and the values those fields need to have when an agreement is in place.
- The Operator Signing Root CA or an Operator Signing Intermediate CA (see clauses 11.3.2 and 11.3.4.5).

Depending on the bilateral agreement, the list may include other information such as the following examples:

- An icon or name for the operator application if the operator application is to appear in a menu provided as part of the terminal UI (e.g. a source selection menu as described in clause 5.2.1).
- A definition of which parts of the manufacturer UI are allowed to be replaced by the operator application (see clause 6.4).
- For IP discovered operator applications, which RF-based channel list is visible to the operator application if the terminal has separate channel lists or operating modes, e.g. a terrestrial operating mode with its own channel list and a satellite operating mode with its own channel list.
- The country or countries or market(s) in which the operator application is deployed.
- Which operator application keys the operator application is permitted to reserve (see clause 10.1.3).

There will additionally be local state that the terminal needs to keep for each installed application.

Some of the information for an entry in the list can be populated when an operator application is discovered (e.g. name, icon). Other information shall be populated up-front (e.g. organisation_id, operator signing CA, discovery mechanism) either at the time a terminal is manufactured or by a software update or by some kind of secure provisioning mechanism.

A terminal supporting privileged operator applications would most likely have more than one entry in this list. A terminal supporting operator-specific operator applications would most likely have only one entry in the list, the operator that it is specific to.

6.6.2 Adding operators and operator applications to terminals

In order to enable adding operators and operator applications to terminals that are already in the market, the following are recommended:

- That terminals support all applicable discovery methods defined in clause 6.1.

NOTE: Discovery based on DVB-SI NIT/BAT would not be applicable to a pure IPTV set-top box. Discovery based on CICAM would not be applicable to terminals without a Common Interface slot.

- That terminals include a secure provisioning mechanism that can add operators to the list of supported operators in clause 6.6.1 including all the associated information needed for discovery and installation.

Operators using different network technologies are likely to use different discovery methods as follows:

- DVB-S/T operators are expected to use one of the NIT/BAT discovery mechanisms.
- DVB-C and IPTV operators are expected to use DNS SRV lookup to a standardized address.

6.6.3 Installed operator applications

A terminal may have multiple installed operator applications. The user will typically be able to switch between them using the source selection menu (see clause 5.2.1) including starting them if they are not running.

NOTE 1: The present document does not require terminals to be able to have multiple operator applications installed at the same time nor does it require support for multiple operator applications to run at the same time.

Some operator applications replace UI elements such as PVR (see clause 6.4) which correspond to either remote control keys or entries in the terminal menus where it would be appropriate to launch the operator application if it is not running. If multiple operator applications are installed which replace the same such UI elements then it is implementation specific which would be launched.

NOTE 2: Where multiple operator applications are installed, particular care is needed if a generic operator application key (see 10.1.3) like EPG or channel list would be used to launch an operator application.

6.7 Removal of operator applications

Once installed, operator applications may be removed by two mechanisms as follows:

- Terminals supporting privileged operator applications shall provide the user with a mechanism to remove them.

NOTE: Since an operator-specific operator application provides the user interface dedicated to that operator, removing one makes no sense. If the user does not wish to receive that operator`s services then the terminal most likely becomes useless.

- Privileged operator applications shall be able to remove themselves using the `opAppUninstall` method defined in clause A.2.2.2.

7 Formats and protocols

7.1 Operator application signalling

7.1.1 Launch and startup context signalling

A terminal may provide multiple entry points to the same operator application from different parts of the terminal UI. Depending on which entry point they use, the user may have different expectations. In these circumstances, the operator application needs to understand the user's expectations in order to be able to present the appropriate starting page. If the terminal provides such different entry points, it shall use table 2a in clause 6.2.2.6.2 of ETSI TS 102 796 [] or the additional values defined in table 10 to provide the `<launch location>` context from where the operator application is launched. An additional `<startup location>` context may be used as defined in table 11, to inform the operator application about the function requested by the user.

If the operator application is not already running, the terminal shall launch the operator application as described in clause 6.3.2.2.

If the operator application is already running (whatever its state), the terminal shall raise an `OperatorApplicationContextChange` event as defined by clause A.2.2.1.

Table 10: Operator application launch locations and values

User interface view or terminal-specific application	<launch location> value
Any part of the terminal's installation screens	`install`
Any part of the terminal's settings screens	`settings`
The terminal's source selection menu	`source`
The terminal resumed from standby	`standby`
Power to the terminal was turned on	`powerup`
The terminal restarted the operator application	`restart`

Table 11: Operator application startup locations and values

User interface view or terminal-specific application	<startup location> value
The user wants to go to TV watching mode. This is the default access method (e.g. if the user selects the operator application in the source selection menu). Typically the operator application will select the last channel watched the previous time it was used and show its normal UI for when a channel is newly selected, e.g. a channel banner.	(startup location not signalled)
An operator application is not running and the user wants to directly access the EPG of the operator application from within a terminal-specific application. Possible access methods are e.g. pressing a GUIDE button on the remote control or selecting a suitable entry in the main menu. When an operator application is running, it can listen for the VK_GUIDE key event, move to the foreground (if needed) and switch to the EPG without this mechanism being used.	`opapp-epg`
The user wants to directly access the library of recorded or downloaded content from within a terminal-specific application (e.g. terminal menu system or PVR button while operator application is not running).	`opapp-pvr`
The user wants to access the settings of an operator application from within a terminals-specific application (e.g. terminal settings menu related to the operators specific settings).	`opapp-settings`
An operator application is not running and the user wants to get a list of the available linear (live) TV channels (services) - without any data about the programmes. When an operator application is running, it can listen for the VK_CHANNELS key event, move to the foreground (if needed) and show the channel list without this mechanism being used.	`opapp-channellist`
The user has opted to install an operator application during installation of a terminal. After the OpApp has been downloaded and the process described in clause 6.1.8 has been followed, depending on the bilateral agreement, the terminal may launch the operator application in an installation context to perform operator-application specific installation. The present document supports operator application specific installation happening either; • Immediately after the end of the end of the terminal's own installation process (using this startup location value) or; • The first time the operator application is run after the end of that process (using the appropriate startup location value for what caused the operator application to be launched). The present document does not support operator application specific installation happening in the middle of the terminal's own installation process.	`opapp-install`
The user triggers an action using the terminal UI that results in an operator application being uninstalled. This may be an explict request to uninstall the operator application using the terminal UI. Alternatively the user may choose a feature from the terminal UI (e.g. channel scan) that implicitly results in the operator application being uninstalled and then re-installed afterwards.	`opapp-uninstall`

User interface view or terminal-specific application	<startup location> value
There is no means for an operator application triggered with this startup location to abort the uninstall process. This context gives the operator application the opportunity to communicate with the user and perform any appropriate administration in the operator's back office systems.	
The user has used the terminal UI to access a specific channel. The terminal selects the channel and then launches the operator application to show its normal UI for when a channel is newly selected, e.g. a channel banner.	`opapp-select-channel`
The user has used the terminal UI to access the operator application from a context where the operator application may want to show a operator specific home screen as an alternative to showing a particular channel.	`opapp-homemenu`
The operator application is involved in retrieving DRM / CA licenses for channels and the terminal is in a state where the operator application is not being presented, but is about to record a programme. The operator application may retrieve all DRM licenses or CA entitlements that are required to decrypt the content.	`opapp-recording`
The user has used the terminal UI to access the operator application search screen.	`opapp-search`
The user makes a choice on the terminal that results in the terminal entering a mode where the operator application is still running but linear TV is no longer active, (e.g. an HDMI input or an application from a global VOD service provider is selected). This context is only valid for an operator application in the background state. An operator application that is in any other state when the user leaves linear TV shall be transitioned to the background state before the context change to opapp-silent is applied.	`opapp-silent`

Further launch location values may be defined in the bilateral agreement.

Examples shown below are when the operator application is not running. Parameters can be placed in any order and shall be supported by the operator application.

EXAMPLE 1: The user has powered on the TV and the terminal launches the operator application to start in TV watching mode using the following url.

`hbbtv-package://456.123/index.html?lloc=powerup`

EXAMPLE 2: The user has powered on the TV and the terminal shows the home menu. The user chooses the operator application on a source selection menu and the terminal launches the operator application to go to TV watching mode.

`hbbtv-package://456.123/index.html?lloc=source`

EXAMPLE 3: The user has powered on the TV and the terminal shows the home menu. The user chooses the EPG of the operator service on the home menu and the terminal launches the operator application to go to the EPG.

`hbbtv-package://456.123/index.html?lloc=homepage&sloc=opapp-epg` or

`hbbtv-package://456.123/index.html?sloc=opapp-epg&lloc=homepage`

EXAMPLE 4: The user has powered on the TV and the terminal shows the home menu, the user speaks to the remote control's microphone and requests the recordings section. The terminal launches the operator application to go to the recordings.

`hbbtv-package://456.123/index.html?lloc=speech&sloc=opapp-pvr` or

`hbbtv-package://456.123/index.html?sloc=opapp-pvr&lloc=speech`

7.1.2 Status launch parameter

The terminal shall use the parameter `status` to provide additional information to the operator application in the form "`status=<status>`" with supported values as defined in table 12. The terminal shall add the status query parameter at the end of the operator application URL, using either a "?" or a "&" character in order to maintain a legal URL structure as defined in IETF RFC 3986 [16]. This status can provide further information to the operator application, whilst retaining the launch and startup contexts of the operator application. If none of the status values in table 12 are applicable, the terminal shall exclude the status parameter from the initial launch of the operator application.

Table 12: Status launch parameter values

Status Description	<status> value
The update of an installed operator application was successfully installed. This value shall only be provided the first time that the operator application is launched after the successful update.	`updateSuccessful`
The update of an installed operator application has failed. This value shall only be provided the first time that the operator application is launched after the failed update attempt.	`updateFailed`
Indicates to the operator application that an error had occurred which caused the application to be terminated.	`error`

EXAMPLE 1: Example usage when the user requests access to operator EPG from the terminal's menu after an unsuccessful update.

```
hbbtv-package://456.1234/index.html?lloc=homepage&sloc=opapp-epg&status=updateFailed
```

EXAMPLE 2: Example usage when the user requests access the operator EPG from the terminal's menu when no status value is applicable.

```
hbbtv-package://456.123/index.html?lloc=homepage&sloc=opapp-epg
```

EXAMPLE 3: Example when the terminal had restarted the operator application following an error that occurred. (e.g. out of memory).

```
hbbtv-package://456.123/index.html?lloc=restart&status=error
```

7.2 Extensions to broadcast signalling

7.2.1 Application overlay descriptor

The broadcast application signalling defined in clause 7.2.3.1 of ETSI TS 102 796 [] shall be extended with the application overlay descriptor as defined in table 13. This descriptor is intended as a last resort in the event of inappropriate use of overlays by an operator and is not a substitute for constructive commercial relationships. For the use of this descriptor, see clauses 9.1.4 and 8.4.1.1.

Table 13: Application overlay descriptor syntax

Syntax	No. of bits	Identifier	Comments/Value
application_overlay_descriptor (){			
descriptor_tag	8	uimsbf	0x73
descriptor_length	8	uimsbf	
id_count	8	uimsbf	N0
for(i=0;i<N0;i++){			
organisation_id	24	uimsbf	
}			
for(i=0;i<N1;i++){			
reserved_future_use	8	uimsbf	
}			
}			

descriptor_tag: This 8 bit integer with value 0x73 identifies this descriptor.

descriptor_length: This 8 bit field indicates the number of bytes following the descriptor length field.

id_count: This 8-bit field identifies the number of organisation_ids listed in this descriptor.

organisation_id: This 24-bit field carries an organisation_id Any privileged operator application with this organisation_id is not permitted to overlay the broadcast service containing this AIT sub-table.

NOTE: The organisation_id is defined as a 32-bit field but the most significant 8 bits are required to be zero.

reserved_future_use: These reserved bytes may be used for future extensions.

If used, this descriptor shall be placed in the "common" (outer) loop of the AIT only.

7.2.2 Application version descriptor

The broadcast application signalling defined in clause 7.2.3.1 of ETSI TS 102 796 [] shall be extended with the application version descriptor as defined in table 14. This descriptor provides the version and minimum version for an operator application that is signalled using a broadcast AIT (see clause 6.1.5.2) to be used as defined in clause 6.1.8.

Table 14: Application version descriptor syntax

Syntax	No. of bits	Identifier	Comments/Value
application_version_descriptor (){			
descriptor_tag	8	uimsbf	0x74
descriptor_length	8	uimsbf	
reserved	1	bslbf	
version	31	uimsbf	
reserved	1	bslbf	
minimum_version	31	uimsbf	

descriptor_tag: This 8 bit integer with value 0x74 identifies this descriptor.

descriptor_length: This 8 bit field indicates the number of bytes following the descriptor length field.

version: This 31-bit field provides the version number for the application.

minimum_version: This 31-bit field provides the minimum version number for the application.

As defined in clause 6.1.8, when installing a software update for an operator application, terminals are required to reject versions lower than the most recent minimum version number provided for that organisation_id and application_id (if any).

NOTE: This allows trials with new versions of an application to revert back to the last stable version at the end while also allowing versions of an operator application found to have security flaws to be blocked.

7.3 Extensions to broadcast-independent application signalling

7.3.1 Minimum application version

When used for operator applications, the broadcast-independent application signalling defined in clause 7.2.3.2 of ETSI TS 102 796 [] shall be extended as follows to enable the signalling of a minimum version for the application. As defined in clause 6.1.8, when installing a software update for an operator application, terminals are required to reject versions lower than the most recent minimum version number provided for that organisation_id and application_id (if any).

NOTE: This allows trials with new versions of an application to revert back to the last stable version at the end while also allowing versions of an operator application found to have security flaws to be blocked.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
          xmlns:hbbtv="urn:hbbtv:opapp_application_descriptor:2017" xmlns:ait="urn:dvb:mhp:2009"
           targetNamespace="urn:hbbtv:opapp_application_descriptor:2017"
           elementFormDefault="qualified" attributeFormDefault="unqualified" >
    <xs:import namespace="urn:dvb:mhp:2009" schemaLocation="oipf/imports/mis_xmlait.xsd"/>

    <xs:complexType name="HbbTVOpAppApplicationDescriptor">
        <xs:complexContent>
```

```
            <xs:extension base="ait:ApplicationDescriptor">
                <xs:sequence>
                    <xs:element name="MinimumApplicationVersion" type="xs:unsignedInt"
                                minOccurs="0" maxOccurs="1" />
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:schema>
```

Here is an example of an XML AIT using this schema to describe version 9053 (decimal) of an application with a minimum version of 9047. The effect of this would be that once version 9053 has been installed, any attempt to install versions 0 to 9046 (inclusive) though a software update will fail.

```
<?xml version="1.0" encoding="UTF-8"?>
<mhp:ServiceDiscovery
    xmlns:mhp="urn:dvb:mhp:2009"
    xmlns:hbb="urn:hbbtv:opapp_application_descriptor:2017"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

    <mhp:ApplicationDiscovery DomainName="operator.example.com">
        <mhp:ApplicationList>
            <mhp:Application>
                <mhp:appName Language="eng">National Cable operator application</mhp:appName>
                <mhp:applicationIdentifier>
                    <mhp:orgId>123</mhp:orgId>
                    <mhp:appId>456</mhp:appId>
                </mhp:applicationIdentifier>
                <mhp:applicationDescriptor xsi:type="hbb:HbbTVOpAppApplicationDescriptor">
                    <mhp:type>
                        <mhp:OtherApp>application/vnd.hbbtv.opapp.pkg</mhp:OtherApp>
                    </mhp:type>
                    <mhp:controlCode>AUTOSTART</mhp:controlCode>
                    <mhp:visibility>VISIBLE_ALL</mhp:visibility>
                    <mhp:serviceBound>false</mhp:serviceBound>
                    <mhp:priority>1</mhp:priority>
                    <mhp:version>9053</mhp:version>
                    <mhp:mhpVersion>
                        <mhp:profile>0</mhp:profile>
                        <mhp:versionMajor>1</mhp:versionMajor>
                        <mhp:versionMinor>4</mhp:versionMinor>
                        <mhp:versionMicro>1</mhp:versionMicro>
                    </mhp:mhpVersion>
                    <hbb:MinimumApplicationVersion>9047</hbb:MinimumApplicationVersion>
                </mhp:applicationDescriptor>
                <mhp:applicationUsageDescriptor>
                    <mhp:ApplicationUsage>
                        urn:hbbtv:opapp:privileged:2017
                    </mhp:ApplicationUsage>
                </mhp:applicationUsageDescriptor>
                <mhp:applicationTransport xsi:type="mhp:HTTPTransportType">
                    <mhp:URLBase>https://operator.example.com/</mhp:URLBase>
                </mhp:applicationTransport>
                <mhp:applicationLocation>opapps/de/v9053.pkg</mhp:applicationLocation>
            </mhp:Application>
        </mhp:ApplicationList>
    </mhp:ApplicationDiscovery>
</mhp:ServiceDiscovery>
```

7.3.2 Operator application images

The `icon@filename` attribute shall contain a URL from which the image may be retrieved. The URL may be a relative URL, resolved relative to the location of the XML AIT file, or an absolute `https:` URL.

NOTE: Although the attribute is named filename, it contains a URL and not a reference to a file within the operator application package.

Terminals may use the reference to the application icon in the XML AIT for an operator application to obtain an icon or icons for use in their UI. If terminals use this reference then they shall retrieve and locally store the image whenever the operator application is installed or updated. Terminals may access the image prior to installation (e.g. to use in a progress dialogue during setup or when offering the user a choice of operator applications or terminal installation modes).

`icon` elements may contain an additional attribute named purpose. The purpose and intended use of such icons may be defined in the bilateral agreement. Terminals should ignore any icon element with a `@purpose` attribute that they do not understand.

Terminals shall cache icons retrieved from URLs for the duration indicated in the `max-age` directive of the `Cache-Control` HTTP response header. If the HTTP response of the icon URL contains an `ETag` header, the terminal shall cache the `ETag` header value and include it in the If-None-Match HTTP header of subsequent image url requests. If the server returns an HTTP status code `304 Not Modified`, the terminal shall use the previously cached icon.

7.4 Operator application ZIP File

7.4.1 Operator application ZIP File Format

The operator application ZIP File shall conform to ISO/IEC 21320-1 [9].

7.4.2 Interoperability Considerations

Some issues can arise with regards to character encodings and length of file names and relative paths of files on terminals. This clause is intended to help operators avoid potential interoperability issues on terminals.

Operators shall restrict file and folder names to characters in the US-ASCII range. Operators shall prohibit or restrict the use of characters as defined in table 15.

Operators shall restrict the total length of path names including slashes, filenames and extensions not to exceed 250 bytes.

Table 15: Permitted, Forbidden and Constrained Characters within an operator application ZIP File

Codepoint or range	Character or Block Name		operator application ZIP File Restrictions
U+0000 - U+001F			prohibited
U+0020	SPACE		Permitted except as the first and last character
U+0021	EXCLAMATION MARK	!	prohibited
U+0022	QUOTATION MARK +	"	prohibited
U+0023	NUMBER SIGN	#	prohibited
U+0024	DOLLAR SIGN	$	prohibited
U+0025	PERCENT SIGN	%	prohibited
U+0026	AMPERSAND	&	prohibited
U+0027	APOSTROPHE	'	prohibited
U+0028	LEFT PARENTHESIS	(	prohibited
U+0029	RIGHT PARENTHESIS	)	prohibited
U+002A	ASTERISK	*	prohibited
U+002B	PLUS SIGN	+	prohibited
U+002C	COMMA	,	prohibited
U+002D	HYPHEN-MINUS	-	allowed
U+002E	FULL STOP	.	Permitted except as the last character
U+002F	SOLIDUS	/	Only permitted as the segment separator in a path name
U+0030 - U+0039	Numeric numbers		allowed
U+003A	COLON	:	prohibited
U+003B	SEMICOLON	;	prohibited
U+003C	LESS-THAN SIGN	<	prohibited
U+003D	EQUALS SIGN	=	prohibited
U+003E	GREATER-THAN SIGN	>	prohibited
U+003F	QUESTION MARK	?	prohibited
U+0040	COMMERCIAL AT	@	prohibited
U+0040 - U+005A	Capital letters		allowed
U+005B	LEFT SQUARE BRACKET	[	prohibited
U+005C	REVERSE SOLIDUS	\	prohibited
U+005D	RIGHT SQUARE BRAKET	]	prohibited
U+005E	CIRCUMFLEX ACCENT	^	prohibited
U+005F	LOW LINE	_	allowed
U+0060	GRAVE ACCENT	`	prohibited
U+0061 - U+007A	Lowercase letters		allowed
U+007B	LEFT CURLY BRACKET	{	prohibited
U+007C	VERTICAL LINE	|	prohibited
U+007D	RIGHT CURLY BRACKET	}	prohibited
U+007E	TILDE	~	allowed
U+007F	DELETE		prohibited

7.4.3 Operator application ZIP File failure conditions

Terminals shall fail the extraction process under the following conditions;

- The operator application ZIP File is not conformant with clause 7.4.1 and clause 7.4.2.
- The extracted file size would exceed the available storage size available on the terminal.
- A checksum error occurs.
- Any other extraction error occurs that would cause any of the files or folders to be incorrectly or incompletely extracted.

Clause 6.1.9 provides guidance for the terminal to handle the above failure conditions.

7.4.4 Application ZIP file contents

The application ZIP file shall contain an index.html in the top level directory. The terminal shall use the initial index.html as defined in clause 6.3.2.2.

The application ZIP package shall include a copy of the XML AIT file in the root folder with a filename `opapp.aitx` if the encrypted operator application package can be discovered through an XML AIT.

The application ZIP package shall include a copy of the application loop entry from the AIT sub-table beginning with the application_identifier() field and ending with the last descriptor() of the loop entry if the encrypted operator application package can be discovered through an AIT. This shall be placed in a file with name `opapp.ait` in the root folder.

NOTE: If the application ZIP package can be discovered through both an XML AIT and an AIT, then both files are found in the root folder of the application ZIP package. Under these circumstances both files are expected to contain semantically equivalent information. Unpredictable behaviour may result if this is not the case.

The application ZIP file may also contain other HTML files, images, style sheets, JavaScript files, application specific configuration files and other application-specific files.

7.5 Extensions to DVB-I signalling

7.5.1 Operator application controlling media presentation

A `RelatedMaterial` element can be used in DVB-I metadata to signal the location of an application associated with the service or service instance. This uses a `HowRelated` element and a `MediaUri` element.

When there is a need to signal an "Operator application controlling media presentation", this shall be signalled in a `RelatedMaterial` element as follows:

- A `HowRelated` element shall be present with an @href attribute set to `urn:hbbtv:dvbi:mediaopapp`
- A `MediaUri` shall be present containing a URI of the form `dvb://system.ait/orgid.appid`

Such a `RelatedMaterial` element indicates that the operator application with organisation_id `orgid` and application_id `appid` has capabilities to present the media of the DVB-I service or service instance. See clause 9.6 for the terminal requirements associated with such signalling. If it is associated with a Service then the application is linked to all ServiceInstances of that Service.

NOTE: Use of "system.ait" in a DVB URI is a variation to the DVB locator syntax defined in TS 102 851 and is used here to reflect that operator applications are referenced from an XML AIT that is not directly linked to any service.

Such a `RelatedMaterial` element:

- may be associated with a broadband-delivered ServiceInstance or a Service containing only broadband-delivered service instances
- is not valid for a broadcast-delivered ServiceInstance or a Service containing broadcast-delivered service instances. Terminals shall ignore the linked application signalling in these cases.
- can be expected to be ignored by terminals that do not support operator applications

More than one "Operator application controlling media presentation" may be linked to a service or service instance. This indicates that all of the listed operator applications have the capability and the permission to present the media for the service or service instance.

7.6 Metadata for operator application created channels

7.6.1 Overview (informative)

Operator applications may support the discovery of IP-delivered services that are in addition to those that the terminal discovers itself. Such operator application created channels (see clause 9.2.2) are created by passing channel metadata to the `ChannelConfig.createChannelList(String)` method and using the resulting `Channel` object(s) in calls to the usual `setChannel()` methods.

Clause 7.6.2 defines the metadata format used by the `ChannelConfig.createChannelList(String)` method.

7.6.2 DVB-I profile

The `ChannelConfig.createChannelList(String)` method shall support XML documents that satisfy the following requirements:

- The XML document is a valid DVB-I Service Discovery document conforming to TS 103 770.
- The document contains one or more Service elements and zero or one LCNTable elements
- The document does not contain RegionList or SubscriptionPackage elements
- The document contains only broadband-delivered ServiceInstance elements

The metadata for each Service should contain the extensions defined in clause 7.5 to indicate that the service is to be an operator application presented channel (see also clause 9.6.2). Terminals are not required to support services relying on native playback of DASH media content referenced directly from DVB-I metadata for Channel objects created by this mechanism.

The `Channel` objects created shall be constructed from the DVB-I Service Discovery XML in accordance with TS 102 796 v1.7.1 clause O.6.2.1.

8 Browser application environment

8.1 Execution model

The HTML 5 specification which is indirectly referenced through ETSI TS 102 796 [] defines the concept of browsing contexts (see https://www.w3.org/TR/html5/browsers.html#windows) - "an environment in which Document objects are presented to the user".

It further defines other related concepts including top-level browsing contexts, nested browsing contexts and auxiliary browsing contexts.

Terminals shall execute operator applications and regular HbbTV® applications in different top-level browsing contexts.

Terminals implementing the secure contexts specification [i.3] shall consider the origin defined in clause 9.4.1 to be potentially trustworthy as defined in that specification.

8.2 DAE specification usage

Clause A.1 defines additional APIs from the OIPF DAE specification beyond those required by ETSI TS 102 796 []. That clause makes some of those additional APIs mandatory depending on a terminal's support for privileged operator applications and for operator-specific operator applications. Some of those additional APIs may be made mandatory as part of a bilateral agreement. Clause A.2 defines modifications to the APIs from the OIPF DAE specification.

8.3 New JavaScript APIs

8.3.1 APIs for access to proprietary functions

8.3.1.1 The ProprietaryFunctionProvider class

This clause is only applicable to operator-specific operator applications (as defined in clause 4.1.2).

The `oipfObjectFactory` shall be extended to support a `createProprietaryFunctionProvider` method with no arguments that shall return an instance of the `ProprietaryFunctionProvider` class.

The `ProprietaryFunctionProvider` class provides generic access to a terminal's proprietary functions that are not themselves part of the present document. These proprietary functions shall either have a namespace, (e.g. identifying the party who defined them) or shall exist in a global namespace. Use of a namespace is recommended to avoid collisions and a reverse DNS convention should be used, e.g. "com.sometvmanufacturer".

`String[] getTVFunctions(String namespace)`		
Description	Gets an array of the names of the proprietary functions available on the terminal. Except for methods in the global namespace, the names in the array shall be prefixed by the namespace of each proprietary function and separated from the name by a "." character. A bilateral agreement may limit access to some specific namespaces to some operator applications and hence prevent access by all others. Proprietary functions that cannot be accessed by the calling operator application shall not be contained in the result of this function.	
Arguments	`namespace`	If empty, all available proprietary functions shall be returned. Otherwise, the returned list will consist of all available proprietary functions filtered such that those and only those elements which are in the provided namespace will be in the list. NOTE: Empty and non-existing namespaces will hence result in an empty list being returned.

`Boolean queryFunction(String namespace, String name)`		
Description	Checks availability of the proprietary function on the terminal. Returns `false` if the function is not available, `true` otherwise.	
Arguments	`namespace`	This parameter limits the query operation to a certain namespace. If it is empty, only functions in the global namespace shall be queried.
	`name`	Name of the function being queried.

`any invokeFunction(String namespace, String name, Object[] arguments)`		
Description	Calls the proprietary terminal function. If the function does not exist in the specified namespace, the TypeError exception shall be thrown. The value returned by this function is the value returned by the proprietary TV function or `null` if that function does not return anything. The returned value shall be a primitive value as defined below.	
Arguments	`namespace`	This parameter limits the invoke operation to the specified namespace. If it is empty, the global namespace shall be used.
	`name`	Name of the function to invoke.
	`arguments`	Arguments that shall be passed to the proprietary TV function. The array shall contain: • any number of primitive values, • objects, • callbacks. The callbacks shall be JavaScript function object. The callback's arguments shall be also primitive values or objects as defined below,

Whenever a proprietary function returns a primitive value, this shall be a value that takes one of the following JavaScript types:

- null,
- boolean,
- number,

- string.

If the type of argument used in the TV proprietary function or its callback is Object, then it shall be one of the following;

- serializable into a JSON string or
- an object defined in the OIPF DAE specification [2] including both objects of classes defined in that specification such as `Channel` and embedded object such the "video/broadcast" object or
- an object defined in TS 102 796 [1] including both objects of classes defined in that specification and embedded objects

Example usage of the proprietary TV function available in 'com.some_tv_manufacturer' namespace.var pfp = oipfObjectFactory.createProprietaryFunctionProvider();

// Check if the dimTheDisplay function exists on the terminal.

```
if (pfp.queryFunction('com.some_tv_manufacturer', 'dimTheDisplay')) {
    // According to the documentation of the terminal manufacturer, the dimTheDisplay
    // function has following parameters:
    // Integer level: discreet value from 0 to 100
    // Object callback: function invoked if the dim operation has finished.
    // It has one Boolean argument indicating if the operation was successful or not.
    function dimResultClbk(success) {
        if (success) {
            console.log('The display has been dimmed');
        } else {
            console.log('The display could not be dimmed');
        }
    }

    try {
        pfp.invokeFunction('com.some_tv_manufacturer', 'dimTheDisplay', [20, dimResultClbk]);
    } catch (error) {
        console.log('The dimTheDisplay function could not be called');
    }
}
```

Example usage of the global proprietary TV function glued to some other terminal API like oipfObjectFactory.

```
function RegisterShimFunction(name) {
    oipfObjectFactory[name] = function() {
        if (!(name in pfp.getFunctions('')) {
            throw Error('Function ' + name + ' is not available');
        } else {
            pfp.invokeFunction('', name, [].slice(arguments));
        }
    };
}

// called by the operator application to register usage of the exemplary dimTheDisplay function
RegisterShimFunction('dimTheDisplay');

// According to the documentation of the terminal manufacturer, the dimTheDisplay
// function has following parameters:
// Integer level: discreet value from 0 to 100
// Object callback: function invoked if the dim operation has finished.
// It has one Boolean argument indicating if the operation was successful or not.
function dimResultClbk(success) {
    if (success) {
        console.log('The display has been dimmed');
    } else {
        console.log('The display could not be dimmed');
    }
}

try {
    oipfObjectFactory.dimTheDisplay(20, dimResultClbk);
} catch (err) {
console.log('The dimTheDisplay function could not be called');
}
```

8.3.1.2 Deprecated functions

The following functions are defined on the global JavaScript context to support proprietary features of the terminal. They are deprecated and optional.

- A function named `getTVProprietaryFunctions` that is identical to the method `getFunctions` defined above except for the name.
- A function named `queryTVProprietaryFunction` that is identical to the method `queryFunction` defined above except for the name
- A function named `invokeTVProprietaryFunction` that is identical to the method `invokeFunction` defined above except for the name.

8.4 Web APIs

8.4.1 Web Notifications

8.4.1.1 Requirements

- Terminals supporting service workers [i.11] shall support the WHATWG version of the Notifications API [25]. Terminals not supporting service workers shall support at least the older W3C version of the Web Notifications specification [8]. Whichever notifications specification is supported, it shall be supported with the following additional requirements:By default the user agent shall grant permission to display these for all origins (see [i.5]) used by operator applications for which it has a bilateral agreement - see clause 4.3 of [8]. Hence the permission to display notifications for those origins shall be "granted". When a non-installed operator application is launched, the domain of the first page shall be automatically enabled for notifications.
- Where the supported specification requires actions, at least 2 of them shall be supported.
- Terminals shall provide the ability for the user to activate a notification which is optional in the Web Notifications specification [8] - resulting in a "`click`" event being fired to the `Notification` object.
- The notification platform used by terminals should support icons and this may be required by the bilateral agreement between the terminal manufacturer and operator(s).

NOTE 1: Agreement would be needed on the size and encoding of any icons.

- If broadcast content is being presented and the current broadcast channel contains an application overlay descriptor (see clause 7.2.1) that lists the organisation_id of the operator application then notifications shall not be shown.

NOTE 2: It is implementation dependent whether notifications are queued or discarded under these circumstances.

- All notifications originating from an operator application shall be discarded when that application exits.
- If a terminal makes the Web Notifications API visible to regular HbbTV® applications and enables regular HbbTV® applications to run when not visible then the terminal shall ensure that attempts to call the API by those applications fail.

NOTE 3: One example of regular HbbTV® applications running when not visible would be terminals running regular HbbTV® applications when an operator application is in the foreground as defined in clause 6.5.3.

The Web Notifications API shall be supported independent of the state of the operator application except for the opapp-silent context where the behaviour of notifications should be addressed in the bilateral agreement .

8.4.1.2 Usage guidelines

Notifications are intended for time-critical messaging from the operator application to the user resulting from previous user actions when an operator application is in the background state. Some examples include the following;

- Reminders set in an EPG which are imminent.
- Previously booked live pay-per-view events that are about to start.
- A subscription about to expire that would impact content currently being presented to the user.
- An imminent recording that will fail without user intervention.

Other messaging from the operator application to the user should happen when the operator application is in the foreground or transient state.

Notifications are not intended for messaging unrelated to previous user actions or that could wait until the operator application is next in the foreground or transient state. Some examples include the following:

- Promoting products or services to the user, either relating to currently presented broadcast content (e.g. advertising) or not (e.g. live pay-per-view events not previously booked by the user)
- A subscription about to expire that does not impact content currently being presented to user

NOTE: If an operator application mis-uses the notification feature when broadcast channels are selected, this may lead to broadcasters restricting that operator's use of overlays and notifications using the mechanism defined in clause 7.2.1.

8.5 APIs defined in ETSI TS 102 796

8.5.1 Modification to terminalChannel

Clause 8.2.5 of ETSI TS 102 796 [] adds a new read-only property, `terminalChannel`, to the `Channel` class. In order to allow an operator application to change the numbering of channels within the terminal channel list, an operator application shall be able to update the value of this property with any change being stored persistently by the terminal. The terminal shall use this new value in all of its UI. Any other channel with a conflicting channel number shall be re-assigned an unused channel number by the terminal.

9 System integration

9.1 Media decoder and tuner resource conflict resolution

9.1.1 Overview (informative)

The present document defines 3 ways in which broadcast video may be presented: by a regular HbbTV® application, by an operator application and the terminal itself.

- Broadcast video presentation by a regular HbbTV® application is defined in ETSI TS 102 796 [] and referenced specifications.
- Broadcast video presentation by the terminal itself is defined in ETSI TS 102 796 [].
- Broadcast video presentation by an operator application is defined in this clause and references. The extent to which the operator application has control over broadcast video presentation depends on which state the operator application is in. In all states, the operator application may have a `BroadcastSupervisor` object, which is defined in clause A.2.5.

9.1.2 Sharing resources for a video/broadcast object

While the operator application is in the background, transient or overlaid transient state, any video/broadcast object in the application shall be in the Unrealized state, as defined in A.2.4.1.

While the operator application is in the foreground or overlaid foreground state, the rules defined in ETSI TS 102 796 [] shall apply to any video/broadcast object in the application with one addition. A video/broadcast object in the operator application shall be able to pre-empt resources from any running regular HbbTV® application, if such application was not killed when the operator application moved to its current state (as defined in clause 6.3.3.2). In order for such pre-emption to occur, the terminal shall kill the regular HbbTV® application if it is using resources that are required for the operator application.

9.1.3 Sharing resources for other media decoders

When the selected channel is an operator application presented channel (see clause 9.9.2), the operator application may present broadband delivered media using the mechanism defined in clause 9.9.

When any other type of channel is selected or no channel is selected, or when any other method of media presentation is attempted by the operator application, the following requirements shall apply:

While the operator application is in the background, transient or overlaid transient state, any attempts by the operator application to start presentation of broadband delivered media shall fail. For the APIs concerned, the error behaviour shall be that defined for the case when resources are not available. If an operator application is in the foreground or overlaid foreground state, is presenting broadband delivered media and transitions to one of the three states listed above then the media presentation shall be stopped. This shall be reported to the operator application as defined for the API concerned when resources are lost.

While the operator application is in the foreground or overlaid foreground state, it shall be able to control media decoders as required by the rules defined in ETSI TS 102 796 [] with one addition. Any request for media decoder resources shall pre-empt any use of those resources by any running regular HbbTV® application, if such application was not killed when the operator application moved to its current state (as defined in clause 6.3.3.2). In order for such pre-emption to occur, the terminal shall kill the regular HbbTV® application if it is using resources that are required for the operator application.

9.1.4 Broadcast video presentation and privileged operator applications

When a privileged operator application is in the foreground or overlaid foreground states, the presentation of broadcast video shall be blocked and the video completely replaced by transparency if any of the following are true:

- The broadcast video is being presented by a video/broadcast object in the operator application and either the width or the height of the video/broadcast object is larger than 1/3 of the logical 1 280 × 720 coordinate system defined in clause 10.2.1 of ETSI TS 102 796 [].
- The broadcast video is being presented by a video/broadcast object in the operator application and the video/broadcast object is in full screen mode.
- The broadcast video is being presented by the terminal or by a regular HbbTV® application.
- If the channel being presented has an application overlay descriptor (see clause 7.2.1) in the "common" (outer) loop of the AIT and that descriptor lists the organisation_id of the operator application.

While video presentation is blocked, broadcast audio shall continue to be presented.

The video/broadcast object shall not change state and no `onSelectedComponentChange` / `SelectedComponentChange` events shall be generated due to blocking starting or stopping.

NOTE: When presentation of video in a video/broadcast object is blocked, replacing it by transparency will result in graphics below that video/broadcast object in the z-order becoming visible.

Blocking of video presentation shall be re-evaluated when any of the following occur:

- either the width and height of the video/broadcast object or full screen mode are changed; or
- when the current channel of the video/broadcast object changes; or
- when the application signalling in the current channel changes; or
- control of video presentation changes between the operator application and the terminal for any reason.

9.2 Channel lists (informative)

9.2.1 Background

HbbTV® terminals maintain a list of broadcast channels that can be accessed by applications. The process by which this is created and maintained is outside the scope of both ETSI TS 102 796 [] and the present document. That process may be manufacturer specific, network/operator specific or some combination of the two. It may involve the terminal doing some sort of frequency scan to discover available multiplexes and services. Alternatively, it may rely on the terminal obtaining a list of multiplexes and services via some other means. Examples of the latter include the following:

- A cable network operator carrying multiple DVB-SI 'NIT other' tables, one of which is selected during installation by the user entering a number provided by the operator when the user signs up with them.
- A satellite or cable operator using a so-called 'homing mux' which carries DVB-SI NIT, BAT and perhaps 'SDT other'.
- A terminal with a CICAM installed may obtain the list of channels from the CICAM NIT using the CICAM Operator Profile resource with profile_type = 1 as defined in clause 14.7 of the CI Plus Specification v1.3.2 [5].

These and other details of the process for creating and maintaining a channel list may form part of a commercial agreement between a network operator and a terminal manufacturer and may be covered by a certification process run by the operator. Such commercial agreements would be similar to but separate and different from the bilateral agreement referred to by the present document.

Terminals may maintain lists of channels for more than one broadcast physical layer, e.g. a terrestrial list and a satellite list. Such terminals may support different operator applications each linked to different broadcast physical layers and/or they may support operator applications with access to a channel list covering more than one broadcast physical layer.

Alternatively operator applications may not be linked to any broadcast physical layer. In the absence of a standardised broadband service discovery mechanism (e.g. DVB-I), the terminal would provide an empty ChannelList to such an operator application (or no ChannelList at all).

EXAMPLE: An operator application with an empty ChannelList would be one where all the content is as explained in item 3 or 4 in clause 9.2.2 below.

9.2.2 Operator applications and channel lists

An operator application can offer the user content from a variety of different sources in its channel list UI as follows:

1) Channels from the terminal channel list.

2) 'Locally defined channels' - broadcast channels that may not be in the terminal channel list but that, except for this, can be used with all the HbbTV® APIs relating to broadcast channels. These are constructed by the operator application using the `createChannelObject` method defined in clause 7.13.1.3 of the OIPF DAE specification [2].

NOTE 1: Although the method `createChannelObject( Integer idType, String dsd, Integer sid )` was designed for one particular way of offering VoD content in cable networks it is not specific to that use-case. An operator application can use it for broadcast DVB-C/S/S2/T/T2 channels that are not in the terminal channel list regardless of the reason for why they are not in that list.

The way in which the operator application obtains the information to create these locally defined channels is outside the scope of the present document. Some examples include the following:

- An operator application running in a terminal with a broadband connection may obtain such information via that route using, for example, XMLHttpRequest calls to a server.
- An operator application running in a terminal without a broadband connection may obtain such information from a file carried in a DSM-CC object carousel in some kind of home channel.
- An operator application running in a terminal without a broadband connection but with a CICAM may obtain such information from the module via the CI Plus 1.4 auxiliary file system or the CI Plus 1.3 SAS resource (see [5] and [7]).

NOTE 2: Neither the present document nor ETSI TS 102 796 [] include a mechanism to allow an operator application to get channel information from the DVB-SI NIT/BAT/SDT other tables in a 'homing mux' (see above). Such a 'homing mux' could however carry a small DSM-CC object carousel containing the information that an operator application would need to create a set of locally defined channels as referred to above. Alternatively, the process for obtaining the channel list could continue to form part of a commercial agreement between the operator and manufacturers.

NOTE 3: Neither the present document nor ETSI TS 102 796 [] include a way for an operator application to permanently merge channels into the terminal channel list.

3) Content that is offered to the consumer in the operator application UI in a way that looks like a TV channel even though the content is not a channel in any technical sense and is presented using the HTML5 media element or the AV control object and not the video/broadcast object. For example:
   - A sequence of VoD content
   - Some live DASH content

NOTE 1: This scenario is distinct from the "operator application created channel" concept described below.

NOTE 2: Such content does not constitute an "operator application presented channel" (see clause 9.9).

4) An operator application created channel. This is an extension of the category immediately above in which:
   - the terminal is aware that such a channel is selected by virtue of `setChannel` being called with a `Channel` object created from metadata supplied by the operator application (see clause A.2.3 and 7.6)
   - application lifecycle signalling for the channel can be provided in the metadata such that the terminal can start regular broadcast-related HbbTV applications for it (see clause 6.5.4)
   - the terminal has sufficient information about the channel to provide appropriate responses to metadata queries from a regular broadcast-related HbbTV application
   - depending on the metadata, the channel may also be an “operator application presented channel” as defined in clause 9.9
   - the channel is not included in the channel list known to the terminal

If a terminal supports the PVR feature then the OIPF DAE specification requires it to support recording and timeshift both of channels in the channel list and of locally defined channels. The PVR feature will not be able to offer timeshift and recording of content that is not a broadcast channel in a technical sense. Operator applications that offer such content to users but describe it as a channel in their UI will need to be careful to avoid causing confusion.

The bilateral agreement may make RF-based channel scans available to operator-specific operator applications using the channel scan mechanism from the OIPF DAE specification [2] - see clause A.1.

9.2.3 CICAM integration

Discovery of operator applications from the CICAM is defined in clause 6.1.3.4.
Download of the XML AIT from the CICAM is defined in clause 6.1.5.1.
Download of the encrypted application package from the CICAM auxiliary file system is defined in clause 6.1.7.4.

Access to the channel list from the CICAM is mentioned in clause 9.2.2. If the CICAM is removed, the channel list should remain unchanged. An operator application relying on the presence of a CICAM needs to be prepared to tolerate the removal of the CICAM including 1) failure to decrypt the channels that the CICAM would decrypt, 2) inability to access the contents of the auxiliary file system and 3) inability to access the SAS resource.

NOTE: Additional detail may be defined in the bilateral agreeement.

9.3 Display model

The display model defined in clause H.1 of the OIPF DAE specification [2] shall also apply to the present document with the following clarifications:

- The main window of an operator application shall be displayed using either the "Platform-specific application graphic plane" or the "DAE application graphic plane" as defined in that clause. Graphics on the main window of an operator application shall always overlay regular HbbTV® applications regardless of which graphic plane is used to display the operator application (see clause 6.3.3.1 for more information).
- The operator application video window shall be displayed using either the "Subtitle plane" or the "DAE application graphic plane" as defined in clause H.1 of the OIPF DAE specification [2]. Graphics on the operator application video Window (see clause 9.9.3) shall always appear behind regular HbbTV® applications regardless of which graphic plane is used.
- Terminals shall support simultaneous display of the operator application video window, the UI of an operator application and a regular HbbTV® application in the transient state as defined in clause 6.3.3.4 of the present document.
- Terminals shall support simultaneous display of the operator application video window and a regular HbbTV application in the background state.
- Broadband media presented by the operator application for an operator application presented channel using the mechanism defined in clause 9.9 shall be displayed on the “Video plane”.

NOTE: If the terminal chooses to use the "DAE application graphic plane" for operator applications then this requirement means two applications overlaid in the same logical plane at the same time.

9.4 URLs

9.4.1 Origin for an installed operator application

The origin (see [i.5]) for resources loaded from an installed operator application shall be as follows:

```
hbbtv-package://appid.orgid
```

where appid and orgid are the values of the application_id and organisation_id from the AIT or XML AIT from which the application was installed. Both shall be encoded as hexadecimal using lower case letters and no extra leading zeros.

This origin shall be used in all cases where a document or resource origin is used in web specifications including but not limited to Cross-Origin Resource Sharing and Web Storage as referenced indirectly via the Web Standards TV Profile [i.2].

9.4.2 Referencing installed operator applications and resources

An operator application shall be able to reference installed operator applications and resources using the hbbtv-package URL scheme defined in clause 9.4.1 as follows:

- hbbtv-package://appid.orgid

- hbbtv-package://appid.orgid/index.html
- hbbtv-package://appid.orgid/applications/my-other-application.html

An operator application shall be able to access installed resources under the orgid that matches its organisation_id and no other orgids. Regular HbbTV® applications shall not be able to access installed operator applications and resources even if they meet this criteria.

EXAMPLE: An operator application with organisation_id 123 and application id 456 can access installed resources under hbbtv-package://789.123 but not under hbbtv-package://456.999.

When loading the HTML pages of an installed operator application, terminals shall use the DOCTYPE to determine the type of documents (HTML or XHTML).

9.5 Access to broadcast carousels

Privileged or operator-specific operator applications shall be able to mount and access broadcast carousels as defined in clause 7.2.5 of ETSI TS 102 796 [] subject to the following constraints:

- Privileged operator applications shall not be able to access any carousel from a service that carries an AIT sub-table with the HbbTV® application_type unless that AIT signals an HbbTV® application (with any application control code) that has the same organisation_id as the operator application.

NOTE 1: Operator-specific operator applications may attempt to mount and access any carousel.

- Attempting to mount or access a carousel shall fail if this cannot be done without disruption to broadcast content that is being presented and without disruption to any regular broadcast-related HbbTV® application that is running.

NOTE 2: In some cases, operator application access to a particular carousel may only be possible on terminals that have more than one tuner and only when a spare tuner is available.

- Access to carousels by an operator application shall in no way restrict the operation of regular broadcast-related HbbTV® applications and their ability to access carousels.

NOTE 3: The ability for operator applications to access carousels may change in real time due to the actions of regular HbbTV® applications. For example, a regular HbbTV® application may change the selected channel, causing a carousel previously used by an operator application to become unavailable.

9.6 Operator applications and DVB-I

9.6.1 Overview (informative)

Annex O of TS 102 796 v1.7.1 defines how DVB-I service discovery can be used with HbbTV and how DVB-I signalling can drive HbbTV application lifecycle. Behaviour is defined for two types of linked application: “Applications with media in parallel”, where the terminal handles the media presentation for the DVB-I service, and “Application controlling media presentation”, where an HbbTV application handles the media presentation.

When HbbTV operator applications as defined in the present document are deployed in conjunction with TS 102 796 v.1.7.1 or later and DVB-I service discovery is used, a third type of linked application is defined, termed here “Operator application controlling media presentation”. This can be used separately or in conjunction with the other types of linked application.

Clause 7.5.1 defines the DVB-I signalling for “Operator application controlling media presentation”.

Clause 9.6.2 defines the terminal requirements for such signalling.

9.6.2 Operator application presentation of DVB-I services

Where a DVB-I service instance or its parent service has one or more linked applications of type “Operator application controlling media presentation” (see clause 7.5.1), then the Channel representing that service instance shall be considered an operator application presented channel as defined in clause 9.9 if any of the specified operator applications are running.

A Channel representing a DVB-I Service shall also be considered an operator application presented channel when the Channel of the currently selected service instance is an operator application presented channel.

9.6.3 Linkage of both regular and operator applications

A DVB-I service may have one or more regular HbbTV applications linked to it as well as one or more operator applications.

If one or more "Operator applications controlling media presentation" are linked together with an "Application controlling media presentation" or "Application with media in parallel", terminals shall still start the "Application controlling media presentation" or "Application with media in parallel" as normal in accordance with TS 102 796 Annex O.

Where an "Application controlling media presentation" is signalled together with one or more "Operator application controlling media presentation", the "Application controlling media presentation" should test for the presence of the operator application and only attempt to present the media itself if the operator application is not running.

NOTE: Signalling both "Application controlling media presentation" and one or more "Operator applications controlling media presentation" for the same service instance requires great care e.g. to consider potential race conditions. Predictable behaviour could require additional requirements to be specified in the bilateral agreement.

9.6.4 Example usage (informative)

9.6.4.1 Example 1

A number of broadband-delivered services are included in a DVB-I service list and are managed by an operator who provides an operator application that performs all the authorization, content protection, media selection and metrics reporting functions necessary to present those services.

The services are signalled with an "Operator application controlling media presentation". Some services additionally have an "Application with media in parallel" providing interactive services.

Terminals supporting the operator application start it automatically whenever the terminal is in the linear channel viewing mode in accordance with the appropriate bilateral agreement.

When one of the broadband-delivered DVB-I services is selected by the user, the terminal knows that the operator application is able to (and permitted to) present the media for the service and the operator application immediately begins the process of doing so. If the service also has an "Application with media in parallel" this is then started by the terminal and proceeds to show a call to action for the interactive features of the channel.

9.6.4.2 Example 2

A number of broadband-delivered DVB-I services rely on an application to perform authorization, content protection, media selection and metrics reporting for the service.

The services are used in a territory where an operator application is deployed for some terminals and not for others.

The services are signalled with a regular HbbTV application linked as an "Application controlling media presentation" and the operator application linked as an "Operator application controlling media presentation".

Terminals that have the referenced operator application installed run that operator application whenever a linear channel is selected in accordance with the appropriate bilateral agreement. When one of the broadband-delivered DVB-I services is selected by the user, the operator immediately begins the process of presenting the media for the service. The terminal begins to start the "Application controlling media presentation". When started, this application checks the `runningOperatorApplication` property of the `Configuration` class. It sees that a suitable operator application is running and does not attempt to present the media for the channel itself.

Terminals that do not have the referenced operator application installed or do not support operator applications at all ignore the operator application signalling and start the "Application controlling media presentation". Once loaded, this application begins the process of presenting the media.

9.7 Parental Controls

9.7.1 General

Privileged or operator-specific operator applications may require the ability to control presentation of media components via blocking and unblocking a channel subject to the bilateral agreement. This may be based on signalling associated with the media or explicit manual input from the end-user independent of the media or a combination of both. A number of variations exist;

- Operator applications may fully replace the terminal's handling of parental access control,
- Operator applications may replace none of the terminal's handling of parental access control (although this clause is irrelevant in those circumstances).
- Operator applications may fully replace the terminal's handling of parental access control for some services and not for others, see clause 5.5.1,
- Operator applications may only partly replace the terminal's handling of parental access control for either for all services or only for some services and not for others, see clause 5.5.1.

NOTE 1: The last of these is complex and risks a poor user experience where the user is asked to enter PIN codes once by the operator application and a second time by the terminal.

Operator applications shall be able to use a combination of the `PlayStateChange` event, `playState` and `playStateError` properties on the broadcast supervisor object to determine the state of the presented channel. The presented channel state would either be presenting (visible) or connecting (blocked/invisible). Operator applications shall be able to use the `blockAV()` and `unblockAV()` methods on the broadcast supervisor as defined in A.2.5 to control the presentation of the channel. Operator applications shall be able to use `ProgrammeChange` events to trigger reading parental ratings signalled in the broadcast and requesting a change to the transient state to give the user the opportunity to over-ride blocking.

Terminals may offer the ability for users to manually lock specific channels. Terminals may implement different blocking methods e.g. the channel is fully blocked, age level is set by the user and the terminal reviews the parental level of the channel and/or the programme being broadcast. Operator applications may replace this.

While a channel is selected whose media is blocked by parental access control for any reason, any corresponding video/broadcast object and/or broadcast supervisor shall be in the connecting state. Media becoming blocked is a transient error as defined by the video/broadcast object state machine and a `PlayStateChange` event with `state` 1 ('connecting') shall be generated. On a video/broadcast object, the `error` argument shall always be 3. On a `BroadcastSupervisor`, the `error` argument shall be 3 when the blocking was due to signalling associated with the media and shall be 13 when the blocking was due to prior explicit manual input from the user independent of the media. Changing to a channel where content is initially blocked is reported as a `ChannelChangeSucceeded` event followed by a transition to the connecting state as described previously and as defined in the video/broadcast object state machine for transient errors.

NOTE 2: Some versions of TS 102 796 [1] defined that changing to a channel where content is blocked right from the beginning is reported as a `ChannelChangeError`. This is incorrect – that would be a permanent error not a transient error and would be followed by a change to the unrealized state. It has been fixed in errata.

Blocked media becoming unblocked is recovery from a transient error as defined by the video/broadcast object state machine and a `PlayStateChange` event with `state` 2 ('presenting') and `error undefined` shall be generated.

9.7.2 Examples (Informative)

Terminals may use the following use cases as guidance, but are not limited to only these examples. Sequence diagrams for some of these examples and other examples are in annex G.1.

EXAMPLE 1: An operator has a bilateral agreement with the manufacturer to always block AV on startup of the terminal prior to starting the operator application. The terminal blocks the AV, sets the `playState` value to `1` (Connecting) and sets the `BroadcastSupervisor.playStateError` value to `3` (parental lock on channel).

EXAMPLE 2: An operator has a bilateral agreement with the manufacturer to only block AV on startup, if the initial service contains a private parental control descriptor and the service is provided by the operator. On such a service, the terminal prior to starting the operator application blocks the AV, sets the `playState` value to `1` and sets the `BroadcastSupervisor.playStateError` value to `3`.

EXAMPLE 3: An operator application on startup reads the `BroadcastSupervisor.playStateError` property. The operator application displays the appropriate PIN control when `playStateError` has a value of `3`. The user enters a valid PIN and the operator application calls `unblockAV()`. The terminal unblocks the channel and raises a `PlayStateChange` event with the `state` value of `2` (presenting). The terminal sets the `playState` property to `2` and `BroadcastSupervisor.playStateError` to `undefined`.

EXAMPLE 4: Prior to changing channel, the operator application evaluates the intended channel and determines the intended channel may contain sensitive content. The operator application calls the `setChannel` method with the argument `blockAV` set to true and listens for the `PlayStateChange` event with `state` value of `1` and the `error` value of `3`. The operator application may query the `BroadcastSupervisor.playStateError` property for additional confirmation. The operator application displays the appropriate PIN control dialogue. The user enters a valid PIN and the operator application calls `unblockAV()`. The terminal unblocks the channel and raises a `PlayStateChange` event with the value of `2`. The terminal sets the `playState` property to `2` and `BroadcastSupervisor.playStateError` to `undefined`.

EXAMPLE 5: The operator application is running in the background, whilst the terminal is presenting a channel. The terminal subsequently changes the presenting channel via a native voice control feature (or other native user interfaces, companion screen channels etc) to another intended channel. The intended channel contains a private parental control descriptor, where the operator application is responsible for the PIN control. The biliteral agreement says that the terminal will block the presentation of media components when this private descriptor is present. The terminal raises a `ChannelChangeSucceeded` event and a `PlayStateChange` event with the `state` value set to 1 and the `error` value set to `3`. The operator application may query the `BroadcastSupervisor.playStateError` property for additional confirmation. The operator application displays the appropriate PIN control. The user enters a valid PIN and the operator application calls `unblockAV()`. The terminal unblocks the channel and raises a `PlayStateChange` event with the value of `2`. The terminal sets the `playState` property to `2` and `playStateError` to undefined.

EXAMPLE 6: The operator application is running and in the background, whilst the terminal is presenting a channel. The terminal raises a `ProgrammesChanged` event on a programme boundary when the following programme starts to be presented. The operator application evaluates the intended programme and determines that the programme may contain sensitive content.The operator application calls the method `blockAV()`. The terminal blocks the presenting media components and raises a `PlayStateChange event` with `state` set to 1 and `error` set to `3`. The operator application may query the `BroadcastSupervisor.playStateError` property for additional confirmation. The operator application displays the appropriate PIN control dialogue. The user enters a valid PIN and the operator application calls `unblockAV()`. The terminal unblocks the channel and raises a `PlayStateChange` event with the value of `2`. The terminal sets the `playState` property to `2` and `playStateError` to `undefined`.

NOTE: The operator application may move the application to foreground and resize the video to prelong the showing of the PIN control beyond the transient timeout period and the application being forced to background by the terminal.

9.8 WebSocket Server and JSON-RPC Implementation

9.8.1 WebSocket server

Terminals shall implement a WebSocket Server for JSON-RPC requests as defined in clause 9.9 of TS 102 796 v1.7.1 [26]. The server shall be available at all times when an operator application is running. The server shall be discoverable by presence of the `json_rpc_server` element in the XML capabilities document exposed to the operator application (see clause 10.2.4.7 of TS 102 796 [1]). This element shall be extended with an additional attribute indicating the supported version of the operator application specification as follows:

```
<json_rpc_server url="ws-url" version="spec-version" opapp_version="spec-version"/>
```

Endpoint URLs shall be static for at least the lifetime of the operator application.

9.8.2 Operator application usage

The present document defines APIs that are expressed as JSON-RPC messages sent over the WebSocket connection defined in clause 9.8.1. TS 102 796 version 1.7.1 and later also define APIs that use JSON-RPC over WebSocket.

Table 16 contains a summary of JSON-RPC API usage:

Table 16: JSON-RPC usage

JSON-RPC APIs	Defined in	Available to	Status
JSON-RPC request capability negotiation	TS 102 796 v1.7.1 clauses 9.9.5 and 9.9.6	Regular HbbTV applications and operator applications	Required
Features of TS 102 796 v1.7.1 and later (e.g. accessibility, voice interaction)	TS 102 796 v1.7.1	Regular HbbTV applications and operator applications	Required if TS 102 796 v1.7.1 or later is supported
Operator application presented channels	Present document clause 9.9.4	Operator application only	Required
Integration between MSE-based media playback and terminal user interface for component selection	Present document clause 10.1.1.6	Operator application only	Required if the terminal UI for component selection is not fully replaced by the operator application and if the operator application presents content using Media Source Extensions [21] that is not associated with an operator application presented channel

9.8.3 API access

Terminals shall ensure that APIs that are restricted to operator applications (see table 16) are not made available to regular HbbTV applications and that attempts by regular HbbTV applications to negotiate their use using the request capability negotiation mechanism defined in clause 9.9.5 of TS 102 796 v1.7.1 shall fail. The following are examples of how this may be achieved:

- By advertising different WebSocket server endpoints to regular HbbTV applications and operator applications (by returning different endpoint URLs in the XML capabilities document exposed to the applications) and ensuring that the endpoints support only the subset of APIs that the application in question is permitted to access.
- By exposing the same WebSocket server endpoint to all applications but arranging that the implementation is aware of the type of the calling application and allowing the restricted APIs to be negotiated, and API calls to succeed, only where the caller is an operator application.

NOTE: There is no requirement for the terminal to enforce separate access restrictions for the operator application video window and the operator application main window but they may do so.

Where different WebSocket server endpoint URLs are used, each shall contain a unique randomly-generated part (see clause 11.7 of TS 102 796).

At a minimum, the terminal shall simultaneously support one WebSocket connection from the main window of an operator application and one connection from the operator application video window. Terminals supporting TS 102 796 v1.7.1 or later shall support these simultaneously with one connection from a regular HbbTV application. Each shall be able to negotiate capabilities independently using the capability negotiation mechanism defined in clause 9.9.5 of TS 102 796 v1.7.1 [26]. With reference to that clause, if JSON-RPC methods are supported for both the operator application main window and the operator application video window, capability negotiation shall be applied separately for each window as if they were separate applications.

9.9 Operator application presented channels

9.9.1 Overview (informative)

Certain types of channels are designated by clause 9.9.2 as "operator application presented channels". For these channels, the terminal does not handle any video or audio presentation directly. Instead, the operator application implements the media presentation for the channel in JavaScript.

Operator applications are able to present media for such channels using HTML5 Video elements within a second window (hereafter the "operator application video window") that is positioned behind the graphics of regular HbbTV applications.

For such channels, operator applications are responsible for:

- opening a document in the operator application video window that will handle video, audio and subtitle playback for operator application presented channels
- arranging necessary communications between the main operator application window and the operator application video window (e.g. using `postMessage`)
- using heartbeat messages and/or other monitoring approaches to detect and recover from error conditions in the operator application video window
- identifying the correct media to present
- obtaining any necessary DRM licences
- checking user preferences and enabling access services where appropriate (e.g. by rendering subtitles, selecting audio description etc.) – see TS 102 796 clause 10.2.7.6.
- minimising memory usage when no operator application presented channel is being presented

NOTE: If this is not done, the likelihood of the terminal needing to freeze or destroy the operator application video window will increase (see clause 9.9.3).

- managing the play state of the video elements such that at most one is playing at any time
- stopping and starting media when requested by the terminal – see clause 9.9.4.5
- positioning and scaling video presentation when requested by the terminal – see clause 9.9.4.5.4
- controlling the relative volume of audio presentation when requested by the terminal – see clause 9.9.4.5.5
- responding when an operator application presented channel has been selected or changed – see clause 9.9.4.5.1
- providing the terminal with media position information by regularly calling `org.hbbtv.ipplayback.mediaPositionUpdate` throughout the period that the channel is being presented – see clause 9.9.4.4.2
- informing the terminal whenever presentation of the channel media starts and stops by calling `org.hbbtv.ipplayback.statusUpdate` – see clause 9.9.4.4.1

- providing synchronisation timeline mappings for Inter-device media synchronisation or Application to media synchronisation – see clause 9.9.4.4.4
- providing information about available media components and responding to component changes requested by the terminal on behalf of the user or on behalf of regular HbbTV applications – see clause 9.9.4.4.3
- monitoring any in-stream or time dependent parental rating or guidance signalling and ensuring that restricted content is not shown without appropriate authorization

9.9.2 Scope

The following channels shall be operator application presented channels:

- DVB-I service instances for which the DVB-I metadata indicates that the operator application is able to handle media presentation (see clause 9.6.2 and 7.5.1)
- Operator application created channels (see clause 9.2.2) for which the DVB-I metadata indicates that the operator application is able to handle media presentation (see clause 7.6)
- Other channel types delivered over IP as specified by the bilateral agreement

NOTE: Terminals can determine at any time whether a given channel is an operator application presented channel or not. This can be achieved either using the idType field of the Channel object (for cases where a whole class of channels falls within scope) or by examining channel metadata for a specific channel.

9.9.3 Operator application video window

Operator applications shall at all times have access to an additional window, termed the operator application video window, into which a document can be loaded to present the media for an operator application presented channel.

A document is loaded into this window (replacing any previously loaded document) as follows:

```
window.open(url, "_opappvideo")
```

where `url` is a URL using the `https:` or `hbbtv-package:` scheme with the same origin as the document calling this method. Terminals are not required to support loading documents from other origins into the operator application video window. Behaviour is undefined if the operator application subsequently transitions to a page from a different origin after instantiating the operator application video window.

NOTE 1: This restriction ensures that the contents of the operator application video window have the same security origin as the main window. As such, they may be able to share common browser processes and resources, minimising terminal memory usage.

The operator application video window shall have the following properties:

- The window shall be visible when (and only when) an operator application presented channel is selected, regardless of the state of the operator application. As an exception, the window may be termporarily hidden if, while an operator application presented channel is selected, video content is being presented that requires subtitle presentation by the terminal (e.g. broadcast subtitles or terminal-rendered TTML subtitles).

NOTE 2: The exception here permits an implementation option in which the same graphics plane and/or graphics memory is used for the operator application video window as for terminal rendered subtitles. An example of a case where the exception would apply is if a regular HbbTV application for an operator application presented channel stops presentation of linear channel video and, while the channel remains selected, presents some VOD content using the terminal's native DASH player and subtitle renderer.

- The window shall be fully transparent until such time as a document is loaded into it.
- The window shall remain visible during a channel change from one operator application presented channel to another.
- The terminal shall be able to take media playback resources away from the window, pausing any HTML5 media elements that are potentially playing, as soon as an operator application presented channel is no longer

selected. Terminals shall not take away such resources during a successful channel change from one operator application presented channel to another.

EXAMPLE: When changing channel from an operator application presented channel to a broadcast channel, the terminal is not required to wait for the operator application to detect the channel change and stop media presentation itself before taking resources to show the terminal-presented broadcast channel.

- The window shall never gain focus and shall not receive any key events. Behaviour is undefined if the key set APIs are called from the operator application video window.
- Communication using `postMessage` shall be supported in both directions between the two windows using the `WindowProxy` objects returned by `window.open()` and `window.opener`. The event message listener shall support the `origin` and `source` properties of the `Message` event.
- The window shall be positioned behind any running regular HbbTV application and hence will also lie behind the main window of the operator application (see also clause 9.3).
- The window shall have the same co-ordinate system as the main window of the operator application but may have a different resolution and colour depth. However, the resolution and colour depth of each window shall comply with the requirements in TS 102 796 clause 10.2.1.
- Unless otherwise stated, the window shall have access to the same APIs as the main window of the operator application.
- Browser stored data (e.g. for features such as `IndexedDB`, `LocalStorage`, caches including `ServiceWorkers`) shall be consistent between the main window and the video window.

NOTE 3: If applications additionally require synchronisation of data between windows, they need to use other mechanisms to achieve this, e.g. using `postMessage`.

- Documents in the operator application video window may include HTML5 `<video>` or `<audio>` elements and may use the Media Source Extensions (MSE). Unless otherwise stated, these shall have all of the capabilities defined in the core HbbTV specification. Limits on the numbers of `MediaSource` and `HTMLMediaElement` objects that may be instantiated concurrently apply separately to the operator application main window and the operator application video window.
- Only one `HTMLMediaElement` from either the operator application main window, operator application video window or a regular HbbTV application can be playing at any one time (see also clause 9.9.6).
- Concerning the requirements in clause 9.6.13 of TS 102 796 relating to "quantities of data" that "MSE SourceBuffers shall be able to accept" "without invoking the coded frame removal algorithm", these shall apply to operator applications only when there is at most one MediaSource object with buffered data in either the operator application main window or the operator application video window. They do not apply if there is more than one MediaSource object with buffered data in the operator application.
- There is no requirement for the terminal to support the instantiation of any `video/broadcast, BroadcastSupervisor, MediaSynchroniser` or `AVControl` object within the operator application video window. If operator applications need to use such objects, they should be placed on the main window of the operator application.
- Use of the JSON-RPC over WebSocket API mechanism shall be supported for the operator application video window. See clause 9.8.

NOTE 4: Typically, the operator application's main window will include logic to handle channel changes and the operator application video window will include logic to handle media operations such as pause, track selection etc.

NOTE 5: The operator application video window will typically be initialised by the operator application shortly after it starts. This allows any time-consuming steps required to initialise a media player (e.g. a Javascript DASH player and subtitle renderer) to take place once. When an operator application presented channel is subsequently selected, the operator application video window can immediately initiate presentation of the channel video, reducing channel change times.

- The window shall be closed when the operator application is destroyed (see clause 6.3.2.3).
- The page loaded in the operator application video window should continue to run at all times while the operator application is itself running to enable rapid channel changes between broadcast channels and operator application presented channels. However, when no operator application presented channel is selected, the terminal may freeze or discard the window to free memory for other purposes. If this action is taken, the terminal is responsible for restoring or reloading the page into the operator application video window when (or ideally before) an operator application presented channel is next selected, waiting for capability negotiation to complete before calling any player methods (see clause 9.9.4). Exceptionally, if memory issues require the operator application video window to be reloaded whilst an operator application presented channel is selected, the terminal may reload the page, starting a new player session once capability negotiation has completed.

  If the operator application video window is restored or reloaded in this way, the terminal shall:

  - restore the video window to the URL it was on when it was frozen, discarded or reloaded,
  - ensure that the `WindowProxy` objects returned by `window.opener` in the operator application video window and by the `window.open()` in the operator application main window continue to refer to the respective windows such that messages may continue to be passed between them using `postMessage`,
  - preserve any data stored in sessionStorage from the time when the page was frozen, discarded or reloaded.

9.9.4 API for operator application presented channels

9.9.4.1 Introduction (informative)

In order to allow operator application presented channels to appear similar to broadcast channels and natively-presented IP channels, both to regular HbbTV applications and to the user, an API is defined by which the terminal (on behalf of the user, a regular HbbTV application or the user interface of an operator application) can control and understand the presentation of linear channel video by the operator application.

When regular HbbTV applications use the `video/broadcast` object to select and control presentation of an operator application presented channel, the terminal uses this API to make corresponding requests to the linear channel player running in the operator application video window. Similarly, the API is also used when the user interface of an operator application uses the `video/broadcast` or `BroadcastSupervisor` object to select, control or monitor presentation of an operator application presented channel.

The API is defined in terms of JSON-RPC methods over a WebSocket communication channel.

9.9.4.2 JSON-RPC usage

9.9.4.2.1 Supported methods

The methods defined in clauses 9.9.4.4 and 9.9.4.5 shall be supported for operator applications. except as defined in clause 10.1.1.6, these are only intended to be used by the operator application video window and should not be used by the operator application main window. Any such use should be highlighted in the bilateral agreement. These methods shall not be accessible to regular HbbTV applications and capability negotiation for these by regular HbbTV applications shall fail.

9.9.4.2.2 Request capability negotiation

The negotiation mechanism defined in clause 9.9.5 of TS 102 796 v1.7.1 [26] shall support the JSON-RPC requests defined in clauses 9.9.4.4 and 9.9.4.5.Terminals shall not send JSON-RPC requests defined in clause 9.9.4.5 unless the operator application video window has indicated support for the request via capability negotiation.

Terminals may assume that the operator application is ready to receive requests to select channels as soon as negotiation has been completed.

9.9.4.2.3 Application and terminal responses

All method calls defined in this section return a `result` or `error` property. These responses shall conform to a schema whose normative definition is found in the electronic attachments – see Annex H of the present document.

The value of the `result` property is an object with the following properties:

- The `method` property is a string value that has the same value as that of the `method` property of the original request.
- Other properties specific to individual methods may also be defined.

A response containing an `error` property means that the application was not able to accept and act on the request. The value of the `error` property is an object with the following properties:

- The `method` property is a string value that has the same value as that of the `method` property of the original request.
- The `code` property is a number as defined in Table 10f of TS 102 796 v1.7.1 [26].
- The `message` property is a string and describes the error.
- The `data` property is optional.

NOTE: It cannot be assumed that any text in the message property will be understandable to the user of the terminal.

The example below illustrates a "Not found" error:

```
{
    "jsonrpc": "2.0",
    "error": {
        "code": -3,
        "message": "invalid session ID",
        "method": "org.hbbtv.ipplayer.stop"
    },
    "id": "2021-04-28T18:50:00Z - 485628"
}
```

9.9.4.3 Concepts

9.9.4.3.1 Channel selection and channel sessions

Operator applications are responsible for the media presentation of an operator application presented channel. Presentation of such channels is initiated and controlled within a 'channel session'.

Whenever an operator application presented channel is newly selected, whether by the user or by a regular HbbTV application or operator application calling the `setChannel()`, `prevChannel()` or `nextChannel()` methods of a `video/broadcast` or `BroadcastSupervisor` object, the terminal requests the operator application start a new 'channel session' using the `org.hbbtv.ipplayer.selectChannel` method (see clause 9.9.4.5.1).

A channel session ends when playback of the channel is stopped for the last time using the `org.hbbtv.ipplayer.stop` method (see clause 9.9.4.5.2). The `org.hbbtv.ipplayer.play` method (see clause 9.9.4.5.3) extends a session that has been stopped but cannot be used if a new session has begun. Only one session can be active at once.

The operator application video window should maintain a WebSocket connection with the terminal at all times while an operator application presented channel is selected so that it can respond appropriately to terminal requests. If the connection is lost for any reason, the operator application should reconnect as soon as possible. The channel may become unresponsive or behave incorrectly if the operator application fails to respond to any terminal requests. If this happens the terminal behaviour is undefined.

9.9.4.3.2 Channel status

A state variable called the operator application channel status is associated with a channel session and is used to inform the terminal when an operator application presented channel's media is being presented by the operator application.

This allows the terminal to communicate appropriate state transitions to applications. The state variable has three states: `Connecting`, `Presenting` and `Stopped`.

When the status is `Connecting`, this indicates that the player is trying to start presenting the content; an error condition may also apply.

When a new channel session is created, the initial channel status is `Connecting`. The `org.hbbtv.ipplayback.statusUpdate` method (see clause 9.9.4.4.1) allows the operator application to transition the channel status to `Presenting` once media playback has started, or to report transient error conditions with the channel status remaining as `Connecting`. The `Stopped` status is used if presentation of the channel has been stopped using the `org.hbbtv.ipplayer.stop` method (see clause 9.9.4.5.2).

Changes to the channel status cause state transitions to occur on `video/broadcast` objects bound to the channel and to `BroadcastSupervisor` objects as specified in clause 9.9.5.

NOTE: The channel status reflects the status of the channel player within the operator application whilst the `video/broadcast` object `playState` reflects the state of that particular `video/broadcast` object. The two may differ. For example, a `video/broadcast` object may be in the `Unrealized` state, not be bound to a channel, while linear channel presentation continues under the control of the terminal with the channel status of the player being `Presenting`. Similarly, if media presentation has been stopped and a `video/broadcast` object is in the `Stopped` state, calling `bindToCurrentChannel` will move the `video/broadcast` object to the `Connecting` state while the player is requested to re-start playback. However, the player may change the channel status directly from `Stopped` to `Presenting`.

9.9.4.3.3 Session identifier

When an operator application creates a session to present a channel, it returns a session identifier (sessionID) value that is then passed in subsequent method calls relating to that channel selection. Session identifiers shall be positive, non-zero integers less than 2^31.

A sessionID is considered valid if it matches a sessionID that has been returned from a call to `org.hbbtv.ipplayer.selectChannel`. A sessionID is current if it refers to the most recently created session.

The terminal and the operator application shall check that the sessionID parameter is current before performing any request.

NOTE: This is to guard against race conditions that may arise if a channel change is initiated at the same time as an application attempts to start, stop, seek or otherwise manipulate linear channel presentation.

9.9.4.4 Operator application video window to terminal JSON-RPC requests

9.9.4.4.1 org.hbbtv.ipplayback.statusUpdate

Sets the operator application channel status (see clause 9.9.4.3.2) for an operator application presented channel session, allowing the terminal to transition the state of any `video/broadcast` objects bound to the channel as well as reporting the transition to any `BroadcastSupervisor` object in use. These requests shall conform to a schema whose normative definition is found in Annex H of the present document.

It is expected that this method will be called when the channel content starts playing, when a transient error condition occurs or clears, and when the content stops at the terminal's request.

An error code representing a transient error may optionally be provided. If so, this shall be used to populate the error parameter of the `video/broadcast` object and `BroadcastSupervisor` `PlayStateChange` events. This method may be called to change the error without changing the status.

A status change from `Presenting` to `Connecting` should only be signalled when a specific error code can be included indicating the cause of the transition. A status change from `Presenting` to `Stopped` should only be signalled after receipt of an `org.hbbtv.ipplayer.stop` request. This method should not be called where the status and error values have not changed.

The following table describes the properties of the `params` object for this method, when those properties are required or optional, and the meaning of the values they can take.

The `sessionID` may refer to a current or previous session. If the `sessionID` is not valid, the method shall fail with error code -3 "Not found".

Property	Required?	Values and their meaning
sessionID	Always required	A number representing an operator application presented channel session to which this status applied (see clause 9.9.4.3.3).
status	Always required	A number that shall take one of the following values: 1 – `Connecting`: media for the current channel is not currently being presented by the operator application but the operator application is attempting to start or resume playback 2 – `Presenting`: the operator application is presenting media for the channel 3 – `Stopped`: media for the current channel is not being presented by the operator application because it was requested to stop
error	Required if an error code applies	If the status has changed due to an error, this property's value is a number that is an error code detailing the type of error. Valid values are those defined for the onChannelChangeError event. If no error has occurred, or a previous error has cleared, this property shall not be included.

NOTE: Error code values that may be applicable to operator application presented channels include 1 (in case of temporary inability to access the stream), 3 (parental lock on channel), 9 (insufficient bandwidth), 11 (insufficient resources) and 100 (unidentified error).

The example below illustrates this request:

```
{
    "jsonrpc": "2.0",
    "method": "org.hbbtv.ipplayback.statusUpdate",
    "params": {
        "sessionID": 298356235,
        "status": 1,
        "error": 3
    },
    "id": "1620296880797"
}
```

The example below illustrates a corresponding response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "method": "org.hbbtv.ipplayback.statusUpdate"
    },
    "id": "1620296880797"
}
```

9.9.4.4.2 org.hbbtv.ipplayback.mediaPositionUpdate

Updates the play position and related parameters for the presentation of an operator application presented channel, allowing the terminal to update the corresponding properties of any `video/broadcast` objects bound to the channel and any `BroadcastSupervisor` object in use. Terminals shall fire any `video/broadcast` and/or `BroadcastSupervisor` `PlayPositionChanged` or `PlaySpeedChanged` events required following the update. These requests shall conform to a schema whose normative definition is found in Annex H of the present document.

It is expected that this method will be called regularly during the presentation of an operator application presented channel when any of the associated parameters change. This may occur even if playback has stalled (e.g. due to changes in `playbackOffset`). `playSpeed` should be updated when the `playbackRate` of the HTMLMediaElement changes.The terminal is not expected to extrapolate the values provided in any way. With the exception of any necessary conversion between milliseconds and seconds, the values provided by the operator application shall be surfaced to the relevant `video/broadcast` and/or `BroadcastSupervisor` directly.

NOTE 1: When reporting routine changes in media position, operator applications should avoid calling this method more frequently than once every 250 ms for performance reasons. More precise synchronisation with the media of an operator application presented channel can be achieved using Inter-device media synchronisation or Application to media synchronisation.

The following table describes the properties of the `params` object for this method, when those properties are required or optional, and the meaning of the values they can take.

Implementations shall be capable of handling position values of up to 48 bits in order to accommodate typical media position values used in live streaming.

NOTE 2: Clause O.6.6 of TS 102 796 v1.7.1 provides mappings for these parameters that are appropriate for MPEG DASH delivered linear services.

If the `sessionID` is not current and valid, the method shall fail with error code -3 "Not found".

Property	Required?	Values and their meaning
sessionID	Always required	A number representing an operator application presented channel session to which this status applied (see clause 9.9.4.3.3).
playPosition	Always required	A number in milliseconds with the semantics defined for the property of the same name in the `video/broadcast` object
playSpeed	Always required	A number with the semantics defined for the property of the same name in the `video/broadcast` object
currentTimeShiftMode	Always required	A number with the semantics defined for the property of the same name in the `video/broadcast` object
playbackOffset	Optional	A number **in milliseconds** with the semantics defined for the property of the same name in the `video/broadcast` object. If the corresponding property of the `video/broadcast` object would have the value undefined then this property is not included, otherwise it is required. Note that the property in the `video/broadcast` object has the unit of seconds. Milliseconds is used in this API for consistency with all parameters expressing media time.
maxOffset	Optional	A number **in milliseconds** with the semantics defined for the property of the same name in the `video/broadcast` object. If the corresponding property of the `video/broadcast` object would have the value undefined then this property is not included, otherwise it is required. Note that the property in the `video/broadcast` object has the unit of seconds. Milliseconds is used in this API for consistency with all parameters expressing media time.

The example below illustrates this request:

```
{
    "jsonrpc": "2.0",
    "method": "org.hbbtv.ipplayback.mediaPositionUpdate",
    "params": {
        "sessionID": 298356235,
        "playPosition": 1671428598361,
        "playSpeed": 1.0,
        "currentTimeShiftMode": 3,
        "playbackOffset": 23714,
        "maxOffset": 7200000
    },
    "id": "4259b574-de85-11ed-8382-9f132b58adf7"
}
```

NOTE 3: The meaning of currentTimeShiftMode 3 is defined in TS 102 796 v1.7.1 clause O.6.6.

The example below illustrates a corresponding response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "method":"org.hbbtv.ipplayback.mediaPositionUpdate"
    },
    "id": "4259b574-de85-11ed-8382-9f132b58adf7"
}
```

9.9.4.4.3 org.hbbtv.ipplayback.setComponents

Updates the set of audio, video and subtitle components that are available in the media of an operator application presented channel, allowing the terminal to update the corresponding properties of any `video/broadcast` objects bound to the channel and any `BroadcastSupervisor` object in use, as well as allowing the terminal to populate any context-sensitive menus for component selection that have not been replaced by the operator application. These requests shall conform to a schema whose normative definition is found in Annex H of the present document.

It is expected that this method will be called after selection of an operator application presented channel when the set of components is first known, as well as whenever the set of available components changes while the channel is selected (e.g. at a period boundary in an MPEG DASH presentation).

NOTE 1: This method may also be used when there is MSE-based media playback in the operator application main window. See clause 10.1.1.6.

Terminals shall support setting at least all of the fields of the `AVVideoComponent`, `AVAudioComponent` and `AVSubtitleComponent` classes for which a mapping for MPEG DASH is defined in clause 8.4.2 of OIPF DAE or clause O.6.3 of TS 102 796 v1.7.1.

NOTE 2: Language codes from DASH content can be passed through unmodified both by the operator application and by the terminal. This may result in regular HbbTV applications seeing 2-character language codes.

The following properties are also defined for use in any of the components:

Component property name	Required?	Component property value
dashRole	Optional	An array of objects with string properties `schemeIdUri` and `value` populated with the attributes of MPEG DASH `Role` elements associated with the component. The `value` property may be omitted if no `Role@value` attribute is present.
dashAccessibility	Optional	An array of objects with string properties `schemeIdUri` and `value` populated with the attributes of MPEG DASH `Accessibility` elements associated with the component. The `value` property may be omitted if no `Accessibility@value` attribute is present.
initiallyActive	Always required	A Boolean value indicating if the component is selected at the time of the call.

If the `sessionID` is current and valid, terminals shall update the set of components that are indicated to applications as active via the `getCurrentActiveComponents` method of the `video/broadcast` object and `BroadcastSupervisor` based on the value of the `initiallyActive` field of each object, firing `SelectedComponentChange` events where appropriate.

The following table describes the properties of the `params` object for this method, when those properties are required or optional, and the meaning of the values they can take.

If the `sessionID` is not current and valid, the method shall fail with error code -3 "Not found".

Property	Required?	Values and their meaning
sessionID	Always required	A number representing an operator application presented channel session to which this status applied (see clause 9.9.4.3.3).
componentList	Always required	Shall be an array of objects. Each object's properties shall be name:value pairs carrying at least the fields required to populate an instance of one of the `AVVideoComponent`, `AVAudioComponent` or `AVSubtitleComponent` classes, including the fields of the base class `AVComponent`. Any field whose value would be `undefined` in the `AVComponent` classes shall be omitted. The class to use shall be chosen based on the `type` field which shall be present in each object. The `componentTag` field shall be present for each component and shall have a unique value among the set of components. For future compatibility, terminals shall ignore any additional properties present that they do not understand.

The example below illustrates this request:

```
{
    "jsonrpc": "2.0",
    "method": "org.hbbtv.ipplayback.setComponents",
    "params": {
        "sessionID": 298356235,
        "componentList": [
            {
                "type": 0,
                "componentTag": 10,
                "encoding": "avc3.640028",
                "encrypted": false,
                "aspectRatio": 1.78,
                "initiallyActive": true
            },
            {
                "type": 1,
                "componentTag": 20,
                "encoding": "mp4a.40.5",
                "encrypted": false,
                "language": "en",
                "audioDescription": false,
                "audioChannels": 2,
                "dashRole": [
                    {
                        "schemeIdUri": "urn:mpeg:dash:role:2011",
                        "value": "main"
                    }
                ],
                "initiallyActive": true
            },
            {
                "type": 1,
                "componentTag": 21,
                "encoding": "mp4a.40.5",
                "encrypted": false,
                "language": "en",
                "audioDescription": true,
                "audioChannels": 2,
                "dashRole": [
                    {
                        "schemeIdUri": "urn:mpeg:dash:role:2011",
                        "value": "alternate"
                    }
                ],
                "dashAccessibility": [
                    {
                        "schemeIdUri": "urn:tva:metadata:cs:AudioPurposeCS:2007",
                        "value": "1"
                    }
                ],
                "initiallyActive": false
            },
            {
                "type": 2,
                "componentTag": 30,
                "encoding": "stpp.ttml.etd1|im1t",
                "encrypted": false,
                "language": "en",
                "hearingImpaired": true,
                "initiallyActive": false
            }
        ]
    },
    "id": "1620296880423"
}
```

The example below illustrates a corresponding response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "method": "org.hbbtv.ipplayback.setComponents"
    },
    "id": "1620296880423"
}
```

After such a call to `org.hbbtv.ipplayback.setComponents`, calling `getComponents(null)` on a `video/broadcast` object bound to the channel would return an `AVComponentCollection` containing one `AVVideoComponent`, two `AVAudioComponent`s and one `AVSubtitleComponent` and `getCurrentActiveComponents(null)` would return an `AVComponentCollection` containing one `AVVideoComponent` and one `AVAudioComponent`.

If the call resulted in the currently selected components changing, appropriate `SelectedComponentChange` events would be generated for `video/broadcast` and `BroadcastSupervisor` objects.

9.9.4.4.4 org.hbbtv.ipplayback.setTimelineMapping

Provides synchronisation timeline mapping information for an operator application presented channel. These requests shall conform to a schema whose normative definition is found in Annex H of the present document.

It is expected that this method will be called in response to an `org.hbbtv.ipplayer.resolveTimeline` request, or if the application wishes to proactively provide a timeline mapping for the channel, e.g. when the content starts playing.

This method informs the terminal that values on the synchronisation timeline represented by `timelineSelector` have a tick rate of `tickRate` Hz and can be generated by adding `offset/1000` to the media timeline of the media resource of the media element identified by `mediaObjectId`. Specifically:

- For inter-device synchronisation:

  `ControlTimestamp.contentTime` = (`HTML5MediaElement.currentTime` (in seconds) + `offset/1000`) * `tickRate`

- For application to media synchronisation:

  `MediaSynchroniser.currentTime` = `HTML5MediaElement.currentTime` (in seconds) + `offset/1000`

The method shall replace any previously defined mapping for the specified `timelineSelector`.

If `mediaObjectId`, `offset` and `tickRate` are not present, any previously defined mapping for the specified `timelineSelector` shall be removed and the timeline shall be considered unavailable for all forms of media synchronisation.

The following table describes the properties of the `params` object for this method, when those properties are required or optional, and the meaning of the values they can take.

If the `sessionID` is not current and valid, the method shall fail with error code -3 "Not found".

Property	Required?	Values and their meaning
sessionID	Always required	A number representing an operator application presented channel session to which this status applied (see clause 9.9.4.3.3).
timelineSelector	Always required	String identifying the type and location of the synchronisation timeline being described.
mediaObjectId	Required to define a mapping	Shall be a string matching the "id" attribute of an HTML5 media element in the operator application video window. This property is omitted if a mapping is being removed. Behaviour is undefined if the HTML5 media element being referenced changes its id or is destroyed while the mapping is in place. Applications should not change the id attribute of a media element previously passed to this method whilst the timeline mapping is in force.
offset	Required to define a mapping	A number representing the offset **in milliseconds** to add to the media timeline of the media resource of the specified media element. This property is omitted if a mapping is being removed.
tickRate	Required to define a mapping	A number representing the tick rate of the timeline in Hz. This property is omitted if a mapping is being removed.

The example below illustrates this request and establishes a timeline mapping:

```
{
    "jsonrpc": "2.0",
    "method": "org.hbbtv.ipplayback.setTimelineMapping",
    "params": {
        "sessionID": 298356235,
        "timelineSelector": "urn:dvb:css:timeline:mpd:period:rel:1000",
        "mediaObjectId": "video-player",
        "offset": 1582340,
        "tickRate": 1000
    },
    "id": "1620296880841"
}
```

The following example removes the timeline mapping that was defined by the previous example message:

```
{
    "jsonrpc": "2.0",
    "method": "org.hbbtv.ipplayback.setTimelineMapping",
    "params": {
        "sessionID": 298356235,
        "timelineSelector": "urn:dvb:css:timeline:mpd:period:rel:1000"
    },
    "id": "1633254000890"
}
```

The example below illustrates a corresponding response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "method": "org.hbbtv.ipplayback.setTimelineMapping"
    },
    "id": "1633254000890"
}
```

9.9.4.4.5 org.hbbtv.ipplayback.setPresentFollowing

Updates the current and next programme metadata of the current channel. The metadata provided allows the terminal to update the corresponding programmes property of the `video/broadcast` and `BroadcastSupervisor` objects, as well as allowing the terminal to populate any 'now and next' user interface elements that have not been replaced by the operator application. These requests shall conform to a schema whose normative definition is found in Annex H of the present document.

It is expected that this method will be called after selection of an operator application presented channel when the programme metadata is first known or when the applicable metadata changes during playback. The metadata provided should relate to the current play position which may be in the past if timeshift is in use.

NOTE: For MPEG DASH streams, operator applications may be able to derive such metadata from the Content Programme Metadata defined in clause 9.1.2 of TS 103 285 v1.3.1. In this case, metadata would be updated whenever the DASH player enters a new `Event` with `schemeIdUri` value of `urn:dvb:iptv:cpm:2014`). Other proprietary metadata sources may also be used.

If the call to this method results in any property of any programme changing, terminals shall generate appropriate `ProgrammesChanged` events for any `video/broadcast` object bound to the channel and any `BroadcastSupervisor` object in use.

The operator application may specify metadata for the current event only (e.g. if no next event is currently known), for the next event only (e.g. if the service has not yet begun), for both current and next events, or for neither (e.g. when previously provided metadata is no longer current and there are no further programmes to describe). If no metadata is available for a channel, this method should not be called.

The following table describes the properties of the `params` object for this method, when those properties are required or optional, and the meaning of the values they can take.

If the `sessionID` is not current and valid, the method shall fail with error code -3 "Not found".

Property	Required?	Values and their meaning
sessionID	Always required	A number representing an operator application presented channel session to which this status applied (see clause 9.9.4.3.3).
current	Optional	Object representing a programme which has already started, but has not yet ended. The object's properties shall correspond to name:value pairs carrying at least the fields required to populate an instance of a `Programme` class. Any field whose value would be `undefined` in the `Programme` class shall be omitted. Collections shall be represented as arrays. This property shall be omitted if the service is not currently broadcasting a programme in the schedule or the current programme can no longer be determined by the operator application. For future compatibility, terminals shall ignore any additional properties present.
next	Optional	Object representing a programme which has a start time that immediately follows the end of the current programme. The object's properties shall correspond to name:value pairs carrying at least the fields required to populate an instance of a `Programme` class. Any field whose value would be `undefined` in the `Programme` class shall be omitted. Collections shall be represented as arrays. This property shall be omitted if there is no programme following the current one in the schedule or the next programme can no longer be determined by the operator application. For future compatibility, terminals shall ignore any additional properties present.

The example below illustrates this request:

```
{
    "jsonrpc": "2.0",
    "method": "org.hbbtv.ipplayback.setPresentFollowing",
    "params": {
        "sessionID": 298356235,
        "current": {
            "name": "News Tonight",
            "description": "Including Sport and Weather. [S]",
            "startTime": 1682373600,
            "duration": 1800,
            "channelID": "dvb://233a..3151",
            "programmeID": "crid://tvchannel.co.uk/episode/b0868x5w",
            "programmeIDType": 0,
            "parentalRatings": []
        },
        "next": {
            "name": "Quiz of the Week",
            "description": "Test your brain with these challenging questions. [S][AD]",
            "startTime": 1682375400,
            "duration": 3600,
            "channelID": "dvb://233a..3151",
            "programmeID": "dvb://233a..3151;1f15",
            "programmeIDType": 1,
            "parentalRatings": []
        }
    },
    "id": "1620296880797"
}
```

9.9.4.5 Terminal to operator application video window JSON-RPC requests

9.9.4.5.1 org.hbbtv.ipplayer.selectChannel

Informs the operator application that a new operator application presented channel has been selected and requests the operator application create a new channel session and start playback. These requests and responses shall conform to a schema whose normative definition is found in Annex H of the present document.

This method shall be called when the terminal selects an operator application presented channel by any means, including:

- selection by the user through a terminal user interface with no regular HbbTV application running
- selection by a regular HbbTV application using the `video/broadcast` object
- selection by the operator application using the `BroadcastSupervisor`
- where the operator application was started in response to the selection of an operator application presented channel. In this case, the method call shall occur after the operator application has started and has completed capability negotiation.
- where the operator application was started on power-up (lloc="powerup") and the terminal has already selected a operator application presented channel. In this case, the method call shall occur after the operator application has started and has completed capability negotiation.

When the method is called, the video and audio decoder resources that are available to the HbbTV implementation shall be available to the operator application for use in presenting a newly selected operator application presented channel unless they are in use by an already-running regular HbbTV application that is permitted to continue running in the context of the newly selected channel.

The method shall not be called whilst any terminal-implemented parental control block remains in effect on the channel.

NOTE: If the terminal has a parental control function that blocks access to channels until a PIN code is entered and this mechanism applies to operator application presented channels, then this method would be called only after the PIN had been successfully entered and the terminal had unblocked the channel.

The operator application is expected to prepare to start presenting the media for the channel and to call the `org.hbbtv.ipplayback.statusUpdate` method when the media starts presenting, or when a transient error condition occurs that prevents media presentation.

The following table describes the properties of the `params` object for this method, when those properties are required or optional, and the meaning of the values they can take.

If the specified channel is not known to the operator application, the method shall fail with error code -3 "Not found".

Property	Required?	Values and their meaning
channelType	Always required	A number representing the type of channel, taken from the corresponding field of a Channel object representing an operator application presented channel.
idType	Always required	A number representing the identifier type for the channel, taken from the corresponding field of a Channel object representing an operator application presented channel.
ipBroadcastID	Always required	A string representing the identifier for the channel, taken from the corresponding field of a Channel object representing an operator application presented channel.

The following table describes the properties of the `result` object of the response message for this method, when those properties are required or optional, and the meaning of the values they can take.

Property	Required?	Values and their meaning
method	Always required	`"org.hbbtv.ipplayer.selectChannel"`
sessionID	Always required	A number representing the session identifier for this channel selection (see clause 9.9.4.3.3). This shall be different from the session identifier for any previous channel selection that has been in effect during the last hour and since the operator application was started. NOTE: In practice, a simple counter starting at 1 when the operator application starts and incrementing with each new channel selection will be sufficient. In the unlikely event that such a counter would exceed the maximum value, it can simply re-start at 1.

The example below illustrates this request:

```
{
    "jsonrpc": "2.0",
    "method": "org.hbbtv.ipplayer.selectChannel",
    "params": {
        "channelType": 0,
        "idType": 50,
        "ipBroadcastID": "tag:rai.it,2019:rai-3"
    },
    "id": "1d7e6dc0-e291-11ed-a888-27cb1882af7d"
}
```

The example below illustrates a corresponding response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "method": "org.hbbtv.ipplayer.selectChannel",
        "sessionID": 298356235
    },
    "id": "1d7e6dc0-e291-11ed-a888-27cb1882af7d"
}
```

9.9.4.5.2 org.hbbtv.ipplayer.stop

Requests the operator application stop presentation of an operator application presented channel, releasing associated resources. These requests shall conform to a schema whose normative definition is found in Annex H of the present document.

This method is called when the terminal needs to stop the presentation of the channel for any reason, including:

- a terminal user interface has been invoked that cannot be presented while linear channel video is playing
- a regular HbbTV application has requested the channel be stopped using the `stop` method of the `video/broadcast` object
- a channel change to a different channel has been initiated
- a broadcast independent application is started, causing the linear channel to be no longer selected
- the terminal enters a mode where linear TV is no longer active (e.g. an HDMI input or an application from a global VOD service provider is selected). Typically, this would be accompanied by a transition to the opapp-silent context.

The operator application is expected to immediately stop the presentation of the linear channel, remove all buffered ranges in MSE SourceBuffers and call the `org.hbbtv.ipplayback.statusUpdate` method when complete.

The operator application should ignore the request if playback has already stopped or is in the process of stopping following a previous stop request. A JSON-RPC response message is still required.

The terminal may take back decoder resources immediately, so the HTMLMediaElement that is presenting the channel may already have been paused by the terminal and its allocated hardware resources released.

The following table describes the properties of the `params` object for this method, when those properties are required or optional, and the meaning of the values they can take.

If the `sessionID` is not current and valid, the method shall fail with error code -3 "Not found".

Property	Required?	Values and their meaning
sessionID	Always required	A number representing the operator application presented channel session to which this status applied (see clause 9.9.4.3.3).

The example below illustrates this request:

```
{
    "jsonrpc": "2.0",
    "method": "org.hbbtv.ipplayer.stop",
    "params": {
        "sessionID": 298356235
    },
    "id": "2021-04-28T18:50:00Z - 485628"
}
```

The example below illustrates a corresponding response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "method": "org.hbbtv.ipplayer.stop"
    },
    "id": "2021-04-28T18:50:00Z - 485628"
}
```

9.9.4.5.3 org.hbbtv.ipplayer.play

Requests the operator application to re-start playback of the currently-selected operator application presented channel. These requests shall conform to a schema whose normative definition is found in Annex H of the present document.

This method is called when the terminal needs to start presentation for any reason, including:

- a regular HbbTV application has requested the channel be presented again by calling the `bindToCurrentChannel` method of the `video/broadcast` object following a previous call to `stop`
- a terminal user interface that could not be presented while linear channel video is playing has been dismissed and linear channel presentation needs to restart
- the operator application has been transitioned from the opapp-silent context and needs to re-start presentation of the same linear channel as was previously being presented

When the method is called, the video and audio decoder resources that are available to the HbbTV implementation shall be available to the operator application for use in presenting the currently-selected operator application presented channel unless they are in use by a regular HbbTV application.

The operator application should ignore the request if playback has already started or is in the process of starting following a previous channel selection or play request. A JSON-RPC response message is still required.

The operator application is expected to prepare to start presenting the media for the channel and to call the `org.hbbtv.ipplayback.statusUpdate` method when the media starts presenting, or when a transient error condition occurs that prevents media presentation.

NOTE: In the event that resources are unavailable (e.g. because a regular HbbTV application is using them), the op app would call `org.hbbtv.ipplayback.statusUpdate` with status = 3 and error = 11.

The following table describes the properties of the `params` object for this method, when those properties are required or optional, and the meaning of the values they can take.

If the `sessionID` is not current and valid, the method shall fail with error code -3 "Not found".

Property	Required?	Values and their meaning
sessionID	Always required	A number representing the operator application presented channel session to which this status applied (see clause 9.9.4.3.3).

The example below illustrates this request:

```
{
    "jsonrpc": "2.0",
    "method": "org.hbbtv.ipplayer.play",
    "params": {
        "sessionID": 298356235
    },
    "id": "2021-04-28T19:21:06Z - 1321"
}
```

The example below illustrates a corresponding response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "method": "org.hbbtv.ipplayer.play"
    },
    "id": "2021-04-28T19:21:06Z - 1321"
}
```

9.9.4.5.4 org.hbbtv.ipplayer.setVideoWindow

Instructs the player to position the video for an operator application presented channel at a specified location. These requests shall conform to a schema whose normative definition is found in Annex H of the present document.

This method is called by the terminal whenever the video size and position needs to change. Examples of when this may occur include:

- A `video/broadcast` object bound to the channel is moved (by means of CSS properties or the object's width, height or fullScreen properties)
- The linear channel video changes from being under the control of the terminal to being under the control of a regular HbbTV application or an operator application
- The linear channel video changes from being under the control of a regular HbbTV application or an operator application to being under the control of the terminal (including where this occurs due to an application exiting)
- The linear channel video is under the control of the terminal and the terminal wishes to show the video at less than full-screen size (e.g. within a programme guide)

The parameters `x`, `y`, `width` and `height` specify a new position **in the co-ordinate system of the operator application**. In the event that a regular HbbTV application is running with a different co-ordinate system to the operator application and changes the video position using the video/broadcast object, the terminal shall convert the `video/broadcast` size and position into the co-ordinate system of the operator application.

No argument equivalent to the `fullScreen` property of the video/broadcast object is included. When a regular HbbTV application or an operator application sets `fullScreen` mode on a `video/broadcast` object, this shall be conveyed by `x` and `y` being zero and `width` and `height` being equal to the horizontal and vertical extent of the operator application graphics co-ordinate system. Use of the Fullscreen API [i.12] by the operator application video window is not recommended.

NOTE: Only configurations where graphics and video occupy the same area of the screen are supported.

Terminals may call this method requesting full screen presentation of video even when the video window is already full screen.

The operator application should respond by resizing re-positioning the HTML5 `video` element being used to render the operator application presented channel, together with any associated subtitels.

The following table describes the properties of the `params` object for this method, when those properties are required or optional, and the meaning of the values they can take.

If the `sessionID` is not current and valid, the method shall fail with error code -3 "Not found".

Property	Required?	Values and their meaning
sessionID	Always required	A number representing the operator application presented channel session to which this status applied (see clause 9.9.4.3.3).
x	Always required	A number representing the new horizontal position in the co-ordinate system of the operator application.
y	Always required	A number representing the new vertical position in the co-ordinate system of the operator application.
width	Always required	A number representing the new horizontal size in the co-ordinate system of the operator application.
height	Always required	A number representing the new vertical size in the co-ordinate system of the operator application.

The example below illustrates this request:

```
{
    "jsonrpc": "2.0",
    "method": "org.hbbtv.ipplayer.setVideoWindow",
    "params": {
        "sessionID": 298356235,
        "x": 0,
        "y": 0,
        "width": 1280,
        "height": 720
    },
    "id": "27bceb08-e29c-11ed-a429-8b90714c6ae3"
}
```

The example below illustrates a corresponding response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "method": "org.hbbtv.ipplayer.setVideoWindow"
    },
    "id": "27bceb08-e29c-11ed-a429-8b90714c6ae3"
}
```

9.9.4.5.5 org.hbbtv.ipplayer.setRelativeVolume

Instructs the player to adjust the relative volume for an operator application presented channel. These requests shall conform to a schema whose normative definition is found in Annex H of the present document.

This method is called by the terminal whenever the relative volume needs to change. Example of when this may occur include:

- The `setVolume` method is called on a `video/broadcast` object bound to the channel
- The linear channel video changes from being under the control of the terminal and being under the control of a regular HbbTV application or an operator application
- The linear channel video changes from being under the control of an application and being under the control of the terminal (including where this occurs due to an application exiting)

The `volume` parameter has the same semantics as the argument of the `setVolume` method of the `video/broadcast` class. When linear channel media is not being controlled by a `video/broadcast` object, `volume` shall be 100.

The operator application should respond by setting the `volume` attribute of the HTML5 video or audio element being used to render the operator application presented channel. The video or audio element volume (`html5_volume`) can be calculated as follows:

If `volume` == 0 then `html5_volume` = 0.0

If `volume` > 0 then `html5_volume` = exp ((`volume`-100)/20)

The following table describes the properties of the `params` object for this method, when those properties are required or optional, and the meaning of the values they can take.

If the `sessionID` is not current and valid, the method shall fail with error code -3 "Not found".

Property	Required?	Values and their meaning
sessionID	Always required	A number representing the operator application presented channel session to which this status applied (see clause 9.9.4.3.3).
volume	Always required	A number with the semantics defined for the parameter of the same name in the `video/broadcast` object's `setVolume` method

The example below illustrates this request:

```
{
    "jsonrpc": "2.0",
    "method": "org.hbbtv.ipplayer.setRelativeVolume",
    "params": {
        "sessionID": 298356235,
        "volume": 100
    },
    "id": "3e6fe363cafd055ffc9af247cb0687c0be433f59"
}
```

The example below illustrates a corresponding response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "method": "org.hbbtv.ipplayer.setRelativeVolume"
    },
    "id": "3e6fe363cafd055ffc9af247cb0687c0be433f59"
}
```

9.9.4.5.6 org.hbbtv.ipplayer.pause

Requests the player to pause playback. These requests shall conform to a schema whose normative definition is found in Annex H of the present document.

This method is called when the `pause` method is called on a `video/broadcast` object bound to an operator application presented channel.

The operator application should respond by pausing media presentation for the operator application presented channel and calling the `org.hbbtv.ipplayback.mediaPositionUpdate` method.

The operator application should ignore the request if playback has already paused or is in the process of pausing following a previous pause request. A JSON-RPC response message is still required.

NOTE: While presentation is paused, the linear channel is still considered to be being presented (and for video services, a still frame will be visible). The channel status does not change but the play speed will be zero.

If playback is paused and the play position becomes outside of the range of the time-shift buffer, the linear channel player running within the operator application video window shall not automatically resume playback. The current playback position shall be maintained unless the terminal requests a change, perhaps acting on behalf of the operator application mian window.

The following table describes the properties of the `params` object for this method, when those properties are required or optional, and the meaning of the values they can take.

If the `sessionID` is not current and valid, the method shall fail with error code -3 "Not found".

Property	Required?	Values and their meaning
sessionID	Always required	A number representing the operator application presented channel session to which this status applied (see clause 9.9.4.3.3).

The example below illustrates this request:

```
{
    "jsonrpc": "2.0",
    "method": "org.hbbtv.ipplayer.pause",
    "params": {
        "sessionID": 298356235
    },
    "id": "cdec492686376842ff3ea3f70b17b3f3993752d4"
}
```

The example below illustrates a corresponding response:

```
{
    "jsonrpc": "2.0",
        "result": {
        "method": "org.hbbtv.ipplayer.pause"
    },
    "id": "cdec492686376842ff3ea3f70b17b3f3993752d4"
}
```

9.9.4.5.7 org.hbbtv.ipplayer.resume

Requests the player to resume playback after a pause. These requests shall conform to a schema whose normative definition is found in Annex H of the present document.

This method is called when the `resume` method is called on a `video/broadcast` object bound to an operator application presented channel.

The operator application should respond by resuming media presentation for the operator application presented channel and calling the `org.hbbtv.ipplayback.mediaPositionUpdate` method.

The operator application should ignore the request if playback has already resumed or is in the process of resuming following a previous resume request. A JSON-RPC response message is still required.

NOTE: While presentation is paused, the linear channel is still considered to be being presented (and for video services, a still frame will be visible). The channel status does not change when playback is resumed but the play speed does change.

The following table describes the properties of the `params` object for this method, when those properties are required or optional, and the meaning of the values they can take.

If the `sessionID` is not current and valid, the method shall fail with error code -3 "Not found".

Property	Required?	Values and their meaning
sessionID	Always required	A number representing the operator application presented channel session to which this status applied (see clause 9.9.4.3.3).

The example below illustrates this request:

```
{
    "jsonrpc": "2.0",
    "method": "org.hbbtv.ipplayer.resume",
    "params": {
        "sessionID": 298356235
    },
    "id": "41f5c039327e943dc4a788c3e42c7a2b1928da6b"
}
```

The example below illustrates a corresponding response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "method": "org.hbbtv.ipplayer.resume"
    },
    "id": "41f5c039327e943dc4a788c3e42c7a2b1928da6b"
}
```

9.9.4.5.8 org.hbbtv.ipplayer.seek

Requests the player to seek to a new position. These requests shall conform to a schema whose normative definition is found in Annex H of the present document.

This method is called when the `seek` method is called on a `video/broadcast` object bound to an operator application presented channel.

The operator application should respond by attempting to seek to the requested position within the media of the operator application presented channel and calling the `org.hbbtv.ipplayback.mediaPositionUpdate` method.

The following table describes the properties of the `params` object for this method, when those properties are required or optional, and the meaning of the values they can take.

Implementations shall be capable of handling position values of up to 48 bits in order to accommodate typical media position values used in live streaming.

If the `sessionID` is not current and valid, the method shall fail with error code -3 "Not found".

Property	Required?	Values and their meaning
sessionID	Always required	A number representing the operator application presented channel session to which this status applied (see clause 9.9.4.3.3).
offset	Always required	A number representing the offset **in milliseconds** with the semantics defined for the parameter of the same name in the `video/broadcast` object's `seek` method. Note that the parameter in the `video/broadcast` object method has the unit of seconds. Milliseconds is used in this API for consistency with all parameters expressing media time.
reference	Always required	A number with the semantics defined for the parameter of the same name in the `video/broadcast` object's `seek` method.

The example below illustrates this request:

```
{
    "jsonrpc": "2.0",
    "method": "org.hbbtv.ipplayer.seek",
    "params": {
        "sessionID": 298356235,
        "offset": 5000,
        "reference": 1
    },
    "id": 1682340564001
}
```

The example below illustrates a corresponding response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "method": "org.hbbtv.ipplayer.seek"
    },
    "id": 1682340564001
}
```

9.9.4.5.9 org.hbbtv.ipplayer.selectComponents

Requests the player to make changes to the set of components selected for presentation. These requests shall conform to a schema whose normative definition is found in Annex H of the present document.

This method is called by the terminal whenever one or more selected components needs to change. Example of when this may occur include:

- One of the `selectComponent` or `unselectComponent` methods is called on a `video/broadcast` object bound to the channel
- The linear channel video changes from being under the control of the terminal and being under the control of an application

- The linear channel video changes from being under the control of an application and being under the control of the terminal (including where this occurs due to an application exiting)
- The linear channel video is under the control of the terminal and the terminal wishes to change the selected components

NOTE: This method may also be used when there is MSE-based media playback in the operator application main window. See clause 10.1.1.6.

The following table describes the properties of the `params` object for this method, when those properties are required or optional, and the meaning of the values they can take.

Each component to be selected is identified from the set of available components (as passed to the terminal using the `org.hbbtv.ipplayback.setComponents` method, see clause 9.9.4.4.3) by its `componentTag` value.

If the `sessionID` is not current and valid, the method shall fail with error code -3 "Not found".

Property	Required?	Values and their meaning
sessionID	Always required	A number representing the operator application presented channel session to which this status applied (see clause 9.9.4.3.3).
videoComponents	Always required	An array of numbers representing `componentTag` values that identify the video components to be selected. If the array is empty, the player is requested to stop presenting any video components. Behaviour is undefined if more than one video component is included.
audioComponents	Always required	An array of numbers representing `componentTag` values that identify the audio components to be selected. If the array is empty, the player is requested to stop presenting any audio components.
subtitleComponents	Always required	An array of numbers representing `componentTag` values that identify the subtitle components to be selected. If the array is empty, the player is requested to stop presenting any subtitle components.

The example below illustrates this request, using components selected from the set illustrated in the `org.hbbtv.ipplayback.setComponents` example (see clause 9.9.4.4.3):

```
{
    "jsonrpc": "2.0",
    "method": "org.hbbtv.ipplayer.selectComponents",
    "params": {
        "sessionID": 298356235,
        "videoComponents": [ 10 ],
        "audioComponents": [ 21, 20 ],
        "subtitleComponents": []
    },
    "id": 1682340878654
}
```

The example below illustrates a corresponding response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "method": "org.hbbtv.ipplayer.selectComponents"
    },
    "id": 1682340878654
}
```

9.9.4.5.10 org.hbbtv.ipplayer.resolveTimeline

Requests the player to provide a mapping from a specific timeline to the play position of the HTML media element being used to present the operator application presented channel. These requests shall conform to a schema whose normative definition is found in Annex H of the present document.

The method is called whenever (i) a synchronisation timeline is used with the “Application to media synchronisation” or “Inter-device media synchronisation” features of TS 102 796 [1], (ii) the timeline relates to a `video/broadcast` object bound to an operator application presented channel, and (iii) the terminal does not have a mapping for that timeline.

The following table describes the properties of the `params` object for this method, when those properties are required or optional, and the meaning of the values they can take.

If the operator application understands `timelineSelector`, it should respond by providing a mapping for the requested synchronisation timeline using the `org.hbbtv.ipplayback.setTimelineMapping` method. Until and unless the operator application does so, the timeline shall be considered unavailable for all forms of media synchronisation (see TS 102 796 [1] clause 13.8.2.2).

If the `sessionID` is not current and valid, or the `timelineSelector` is not known to the player, the method shall fail with error code -3 "Not found".

Property	Required?	Values and their meaning
sessionID	Always required	A number representing the operator application presented channel session to which this status applied (see clause 9.9.4.3.3).
timelineSelector	Always required	A string representing the timeline selector to resolve.

The example below illustrates this request:

```
{
    "jsonrpc": "2.0",
    "method": "org.hbbtv.ipplayer.resolveTimeline",
    "params": {
        "sessionID": 298356235,
        "timelineSelector": "urn:dvb:css:timeline:mpd:period:rel:1000"
    },
    "id": 1682341366500
}
```

The example below illustrates a corresponding response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "method": "org.hbbtv.ipplayer.resolveTimeline"
    },
    "id": 1682341366500
}
```

9.9.5 Regular application control of media presentation

This clause defines the behaviour of the `video/broadcast` object when an operator application presented channel is selected. It applies when a `video/broadcast` object is not in the `Unrealized` state and an operator application presented channel is selected. This can arise if a regular HbbTV application or operator application creates a `video/broadcast` object and calls `bindToCurrentChannel()` when an operator application presented channel is already selected. It can also arise if a regular HbbTV application or operator application has a video/broadcast object and tunes to an operator application presented channel using the `setChannel()`, `prevChannel()` or `nextChannel()` methods.

NOTE 1: The `video/broadcast` object can be used by operator applications in the foreground and overlaid-foreground states (see clause 9.1.2) as well as by regular HbbTV applications. One situation where an operator application may wish to use the `video/broadcast` object even though the operator application is itself presenting the media for an operator application presented channel is to resize or position the video, e.g. while a full-screen user interface such as a programme guide is displayed.

`video/broadcast` object state transitions shall be unchanged from those specified in TS 102 796 except in the following cases:

Old state	New state	Behaviour
Connecting	Presenting	This shall occur when the `video/broadcast` object is in the `Connecting` state and the operator application changes the operator application channel status to `Presenting` by calling `org.hbbtv.ipplayback.statusUpdate` If a `ChannelChangeSucceeded` event has not previously been sent since the operator application presented channel was first selected, it shall be sent prior to the `PlayStateChange` event for this transition.
Connecting	Connecting	This shall occur when the operator application's channel status is set to `Connecting` and an error parameter was present in the call to `org.hbbtv.ipplayback.statusUpdate`. Additionally, it may occur if a transient error condition occurs before the operator application video window is asked to present the channel (e.g. if the channel is blocked by terminal parental access controls). If a `ChannelChangeSucceeded` event has not previously been sent since the operator application presented channel was selected, it shall be sent prior to the `PlayStateChange` event for this transition.
Presenting	Connecting	This shall occur when the operator application changes the operator application channel status to `Connecting`, typically an error parameter will also have been present in the call to `org.hbbtv.ipplayback.statusUpdate`.

NOTE 2: Only transitions between the `Connecting` and `Presenting` states have specific behaviour defined for operator application presented channels, based on changes to the operator application channel status. All other transitions occur as specified in TS 102 796.

NOTE 3: When `bindToCurrentChannel` is called on a `video/broadcast` object that is in the `Stopped` state, the state initially changes to `Connecting` whilst the operator application prepares to restart playback. The subsequent transition to `Presenting` occurs as described above.

NOTE 4: When `bindToCurrentChannel` is successfully called on a `video/broadcast` object that is in the `Unrealized` state where the linear channel is already being presented under the control of the terminal, TS 102 796 specifies that the state transitions directly to `Presenting`. The operator application will not be aware of this transition as it relates only to the state of the specific `video/broadcast` object concerned and does not affect the actual presentation of linear channel media.

`video/broadcast` object properties and method calls shall behave as defined in TS 102 796 [1] except that whenever the `video/broadcast` object is in the `Connecting` or `Presenting` state and an operator application presented channel is currently selected, the properties and methods of the `video/broadcast` object shall behave as follows:

Property or method	Behaviour
width height setFullScreen()	Shall cause the `org.hbbtv.ipplayer.setVideoWindow` method to be called with the new video window co-ordinates. See clause 9.9.4.5.4.
playState	As TS 102 796 [1]. State transitions between `Connecting` and `Presenting` shall be as specified above.
bindToCurrentChannel()	As TS 102 796 [1]. Where this requires the media presentation be restarted (i.e. if the video/broadcast object was in the Stopped state), the `org.hbbtv.ipplayer.play` method shall be called. See clause 9.9.4.5.3. Where bindToCurrentChannel is used from the Unrealized state to take control of media that is already being presented by the terminal, no call to the operator application occurs (see NOTE 4 above).
setVolume(volume)	Shall cause the `org.hbbtv.ipplayer.setRelativeVolume` method to be called. See clause 9.9.4.5.5.
getVolume()	This shall return the volume last set by a call to `setVolume` on this `video/broadcast` object, or 100 if no such call has been made for the current channel session.
release()	In addition to the requirements of TS 102 796 [1], the terminal shall report any resulting changes to the video size and position or the audio relative volume by calling the calling the `org.hbbtv.ipplayer.setVideoWindow` and/or `org.hbbtv.ipplayer.setRelativeVolume` methods. See clauses 9.9.4.5.4 and 9.9.4.5.5.
stop()	Shall cause the `org.hbbtv.ipplayer.stop` method to be called. See clause 9.9.4.5.2 Any hardware resources that were being used by the operator application to present media for the channel shall immediately be available if the terminal requires them for another use, e.g. a regular HbbTV application. The terminal does not have to wait for the operator application to respond to the `org.hbbtv.ipplayer.stop` method before taking such resources.
playbackOffset	Shall return the value of `playbackOffset` most recently set by a call to `org.hbbtv.ipplayback.mediaPositionUpdate` or `undefined` if no such call has been made for the current channel session. See clause 9.9.4.4.2.
maxOffset	Shall return the value of `maxOffset` most recently set by a call to `org.hbbtv.ipplayback.mediaPositionUpdate` or `undefined` if no such call has been made for the current channel session.See clause 9.9.4.4.2.
recordingState	Shall have the value 0.
playPosition	Shall return the value of `playPosition` most recently set by a call to `org.hbbtv.ipplayback.mediaPositionUpdate` or 0 if no such call has been made for the current channel session. See clause 9.9.4.4.2.Whenever this value changes, a `PlayPositionChanged` event shall be generated.
playSpeed	Shall return the value of `playSpeed` most recently set by a call to `org.hbbtv.ipplayback.mediaPositionUpdate` or 1.0 if no such call has been made for the current channel session. See clause 9.9.4.4.2. Whenever this value changes, a `PlaySpeedChanged` event shall be generated.
playSpeeds[]	Shall be undefined.
timeShiftMode	Shall return 0. Setting shall have no effect.
currentTimeShiftMode	Shall return the value of `currentTimeShiftMode` most recently set by a call to `org.hbbtv.ipplayback.mediaPositionUpdate` or 0 if no such call has been made for the current channel session. See clause 9.9.4.4.2.
recordNow()	Optional. If present, shall have no effect and return null.
stopRecording()	Optional. If present, shall have no effect and return null.
pause()	Shall cause the `org.hbbtv.ipplayer.pause` method to be called. See clause 9.9.4.5.6. Shall return true.
resume()	Shall cause the `org.hbbtv.ipplayer.resume` method to be called. See clause 9.9.4.5.7. Shall return true.
setSpeed()	Shall have no effect and return `false`.
seek()	Shall cause the `org.hbbtv.ipplayer.seek` method to be called. See clause 9.9.4.5.8. Shall return `true`.
stopTimeshift()	Shall behave identically to `seek(0, POSITION_END)` and return `true`.

programmes	Shall return programme metadata most recently set by a call to `org.hbbtv.ipplayback.setPresentFollowing`. See clause 9.9.4.4.5. The `ProgrammeCollection` shall be of length 0, 1 or 2 and shall include metadata for all programmes (current and next) for which metadata has been provided for the current channel session. If current metadata is available, this shall appear first. The `getSIDescriptors` method of Programme objects shall return `undefined`.
getComponents()	Shall return the list of components of the appropriate type(s) as most recently set by a call to `org.hbbtv.ipplayback.setComponents` or undefined if no such call has been made for the current channel session. See clause 9.9.4.4.3.
getCurrentActiveComponents()	Shall return a subset of the components that would be returned by `getComponents()`. Only components that are active shall be included. If no component list has been set, `undefined` shall be returned.
selectComponent()	Shall make a component selection from the set of components most recently set by a call to org.hbbtv.ipplayback.setComponents. The component selection shall be performed as required for the appropriate method signature (see OIPF DAE clause 7.16.5.1.3). The terminal shall call the org.hbbtv.ipplayer.selectComponents method with the new list of selected components. See clause 9.9.4.5.9. When this causes a change in the set of active components, a SelectedComponentChange event shall be generated.
unselectComponent()	Shall make a component selection from the set of components most recently set by a call to org.hbbtv.ipplayback.setComponents. The component selection shall be performed as required for the appropriate method signature (see OIPF DAE clause 7.16.5.1.3). The terminal shall call the org.hbbtv.ipplayer.selectComponents method with the new list of selected components. See clause 9.9.4.5.9. When this causes a change in the set of active components, a SelectedComponentChange event shall be generated. The unselectComponent(Integer componentType) variant results in no component of the requested type being selected whereas the unselectComponent(AVComponent component) variant results in the terminal choosing a default component from those available.
addStreamEventListener()	Optional. If present, no event listener shall be called.
removeStreamEventListener()	Optional.

The `org.hbbtv.ipplayer.setVideoWindow` method shall be called whenever there is a change to the position or size of a `video/broadcast` object bound to an operator application presented channel. See clause 9.9.4.5.4.

When a `video/broadcast` object is in the `Connecting` or `Presenting` state and an operator application presented channel is selected, then when the "Application to media synchronisation" or "Inter-device media synchronisation" features of TS 102 796 [1] are used, the terminal shall be able to use a synchronisation timeline mapping provided by the operator application to perform synchronisation to the operator application's HTML5 media element (see clause 9.9.4.4.4). If no such mapping has been provided, the `org.hbbtv.ipplayer.resolveTimeline` method shall be called.

9.9.6 Resource management considerations

As described in clause 9.9.1, operator application presented channels are implemented using HTML5 media elements. Since the resource management behaviour for media elements differs from that of broadcast channel media under the control of the terminal or a video/broadcast object, applications (including regular HbbTV applications) may experience minor differences in behaviour when an operator application presented channel is selected. Specifically, the behaviour will be different on a single-decoder terminal if an application attempts to use an HTML5 media element to play content while linear channel content is being presented.

For a broadcast channel presented under the control of the terminal or a video/broadcast object, no HTML5 media element can play on a single-decoder terminal until the broadcast channel presentation is explicitly stopped. By contrast, the linear channel content for an operator application presented channel will pause automatically if another HTML5 media element plays due to the resource management rules for media elements defined in clause 9.6.2 of TS 102 796 [1].

In most cases, applications manage decoder resource constraints by stopping presentation of one type of content before starting another. As such, these differences in behaviour will not generally be noticed by well-authored applications.

NOTE: Operator applications and regular HbbTV applications should not rely on the ability or inability of an HTML5 media element to pre-empt linear channel media presentation, as the behaviour will differ between broadcast channels and operator application presented IP channels.

10 Capabilities

10.1 Terminal capabilities and functions

10.1.1 Component selection

10.1.1.1 Introduction

Default component selection by the Terminal and additional component selection by regular HbbTV® applications is defined in clause 10.2.7 of ETSI TS 102 796 []. The present document specifies the different methods of component selection by operator applications, in line with the approach defined in clause 10.2.7 of ETSI TS 102 796 [].

To meet the different use cases and technical restrictions, this clause defines three methods of component selection that are available to operator applications:

1) The operator application may influence the default component selection mechanism of the terminal by setting the preferred languages for audio and subtitle streams and by enabling/disabling subtitles and audio descriptions. This method is available in all states and for all content delivery mechanisms. The operator application may provide a means for the user to select their preferences in a general setup menu or directly in relation to the currently presented content. See clause 10.1.1.2.

2) The operator application may provide direct component selection for the currently presented broadcast content by using a `BroadcastSupervisor` object. This method is available in all states. It may be used for selecting content-specific broadcast streams such as audio commentary, camera angle, enhanced audio formats, or channel-specific user choices in the absence of a regular HbbTV® application. See clause 10.1.1.3.

3) If the operator application is in the Foreground state or Overlaid foreground state, it may provide direct component selection by using the appropriate methods on the selected media object/element. See clause 10.1.1.4.

10.1.1.2 Component selection via user preferences

An operator application shall be able to set any of the following properties of the Configuration class as defined in clause A.1:

- `preferredAudioLanguage`, as defined in clause 7.3.2 of the OIPF DAE specification [2].
  When the `preferredAudioLanguage` property is set to a new value, all subsequent reads of either `preferredAudioLanguage` or `preferredAudioLanguage47` (where supported) by either this operator application or a regular HbbTV application shall return the corresponding encoding of the new language until either the property is updated again or the operator application terminates. Persistence of the newly set value after the operator application terminates may be addressed in the bilateral agreement.

- `preferredSubtitleLanguage`, as defined in clause 7.3.2 of the OIPF DAE specification [2].
  When the `preferredSubtitleLanguage` property is set to a new value, all subsequent reads of either `preferredSubtitleLanguage` or `preferredSubtitleLanguage47 (where supported)` by either this operator application or a regular HbbTV application shall return the corresponding encoding of the new language until either the property is updated again or the operator application terminates. Persistence of the newly set value after the operator application terminates may be addressed in the bilateral agreement.

- `audioDescriptionEnabled`, as defined in clause A.2.20 of ETSI TS 102 796 [].

- `subtitlesEnabled`, as defined in clause A.2.20 of ETSI TS 102 796 [].

If an operator application changes any of these four properties, component selection for the affected component type(s) shall revert to the control of the terminal. The terminal shall immediately use the updated property values to re-evaluate the default component of the affected types (as defined in clause 10.2.7.2 of ETSI TS 102 796 []) and modify the set of selected components as appropriate.

NOTE: This is the only mechanism that permits an operator application to influence component selection in broadband delivered content presented by a regular HbbTV® application.

10.1.1.3 Direct component selection via BroadcastSupervisor class

In all operator application states, the operator application shall be able to directly select components of the currently presented broadcast content using the method `selectComponent` of the BroadcastSupervisor class as defined in clause A.2.5.

The terminal shall maintain such changes made by an operator application until one of the following occurs:

- The operator application terminates:
  - in which case component selection for that component type shall revert to the control of the terminal;
- The operator application makes a further change:
  - in which case the behaviour shall be as defined by the API where that change was made;
- The operator application calls `unselectComponent`:
  - in which case component selection for that component type shall revert to the control of the terminal;
- The operator application changes any of the user preferences as defined in the previous clause:
  - in which case the requirements of the previous clause apply;
- The user makes a change using the terminal's subtitle/audio description (or other) selection mechanism:
  - in which case component selection for that component type shall revert to the control of the terminal;
- The broadcast channel is changed by any means:
  - in which case component selection for that component type shall revert to the control of the terminal
- A regular HbbTV® application selects a component of the same type:
  - in which case clause 10.2.7.3 of ETSI TS 102 796 [] shall apply with the clarifications defined in clause 10.1.1.5.

NOTE: There is no equivalent class for broadband delivered content. Hence direct component selection is not available for broadband-delivered content presented by a regular HbbTV® application.

10.1.1.4 Standard direct component selection

If the operator application is in the Foreground state or Overlaid foreground state, the operator application shall be able to use all standard methods of direct component selection for content that it is presenting itself as defined in clause 10.2.7.3 of ETSI TS 102 796 [].

The terminal shall maintain such changes made by an operator application until one of the following occurs:

- Any of the events defined in clause 10.2.7.3 of ETSI TS 102 796 [] occurs in regards to the operator application:
  - in which case the behaviour shall be as defined in that clause;
- The operator application changes any of the user preferences as defined in clause 10.1.1.2:
  - in which case the requirements of that previous clause apply.

10.1.1.5 Clarification of component selection by regular HbbTV® applications

Clause 10.2.7.3 of ETSI TS 102 796 [] defines that a terminal shall maintain a change in component selection which was requested by a regular HbbTV® application until one of a set of events occurs. The following event shall be added to that set:

- An operator application makes a further change:
  - in which case component selection for that component type shall be under the control of the operator application.

10.1.1.6 Integration between MSE-based media playback and terminal user interface for component selection

Terminals will likely include a user interface for audio and subtitle component selection. Examples of this include;

- Selection of audio language
- Selection of accessible audio (e.g. audio description, clean audio, dialogue enhancement)
- Selection of NGA preselections
- Selection between multi-channel audio and stereo
- Selection of subtitle language
- Selection of subtitles for specific purposes (e.g. table 2 "subtitle purpose" of EN 303 560 [i.13])

This may be replaced by operator applications as part of UI_TVMODE but this cannot be guaranteed under all circumstances. These menus may be contextual, i.e. the user is offered a choice limited to the components actually available at that specific time in the actual content being played.

The following applies if the terminal UI for component selection is not fully replaced by the operator application and if the operator application presents content using Media Source Extensions [21] that is not associated with an operator application presented channel.

- Terminals shall support a WebSocket server as defined by clause 9.8.
- The server shall be discoverable by the main window of the operator application.
- Terminals shall support JSON-RPC negotiation of the methods `org.hbbtv.ipplayback.setComponents` and `org.hbbtv.ipplayer.selectComponents` from the operator application main window.
- The following shall apply when the main window of the operator application is presenting video using MSE;
  - The JSON-RPC request `org.hbbtv.ipplayback.setComponents` defined in clause 9.9.4.4.3 shall be supported from the main window of the operator application to the terminal.
    - When the operator application uses MSE to present media that is not part of an operator application presented channel, it should provide the terminal with information about the available components using the `org.hbbtv.ipplayback.setComponents` request with a `sessionID` of zero.
    - The method shall not return an error due to the `sessionID` being zero when called from the operator application main window.
    - Operator applications should call this method i) when MSE playback starts, ii) each time the set of available media components for MSE-presented content changes and iii) at the end of a period of presenting content with MSE, passing an empty `componentList` in the latter case.
  - The JSON-RPC request `org.hbbtv.ipplayer.selectComponents` defined in clause 9.9.4.5.9 shall be supported from the terminal to the main window of the operator application.

- Component selections that do not relate to the media of an operator application presented channel shall be signalled with a `sessionID` of zero and shall only be sent to the operator application main window.
- Operator applications should respond to this request by selecting the requested components for presentation and shall not return an error due to `sessionID` being zero when the request is received by the operator application main window.

10.1.2 Minimum terminal capabilities

Minimum terminal capabilities which shall be available to applications are listed in table 16 for general capabilities. Additional capabilities shall be signalled as defined in clause 10.1.4.

Table 16: Minimum terminal capabilities

	Value for regular operator applications	Value for privileged operator applications	Value for operator-specific operator applications	Additional information
PVR management	Unchanged from ETSI TS 102 796 []	Unchanged from ETSI TS 102 796 []	If the PVR feature is supported, the `manageRecordings` attribute of the recording capability shall have the value "`all`".	See clause 9.3.3 of the OIPF DAE specification [2].

10.1.3 User Input

For Privileged and Operator-specific operator applications, the rules governing access to key events differ from those of regular HbbTV® applications and table 17 takes precedence over clause 10.2.2.1 of ETSI TS 102 796 []. The Type column defines when an operator application receives key events for the associated keys. The following paragraphs apply to both privileged and operator-specific operator application unless explicitly stated.

Operator applications shall be able to request key events using the KeySet API as defined in clause 7.2.5 of the OIPF DAE specification [2]. It shall be possible to request key events which are not represented by any of the constants defined in clause 7.2.5.1 of [2] by using the argument `otherKeys` in method `KeySet.setValue`. Subject to the bilateral agreement this may include:

- For both privileged and operator-specific operator applications: Operator application keys as defined in table 17.
- Only for operator-specific operator applications: System keys as defined in table 17 and any other system keys (e.g. as defined in clause 6.2 of OIPF volume 5a [i.2]).

The distribution of key events to the operator application shall depend on the operator application state according to the following rules:

- The following shall apply when an operator application is running in either foreground state or transient state as defined in clauses 6.3.3.2 and 6.3.3.4:
  - When an operator application has input focus, it shall receive key events for the regular HbbTV application keys and operator application keys listed in table 17 that it has requested to receive using the KeySet API.
- The following shall apply when an operator application is running in overlaid foreground state or overlaid transient state:
  - An operator application shall have access to regular HbbTV application keys and operator application keys if these are not taken by the terminal UI that appears on top of the operator application.
- The following shall apply when an operator application is running in the background state:

- If a regular HbbTV® application has input focus, the operator application shall receive operator application keys and regular HbbTV application keys that it has requested to receive using the KeySet API except for those regular HbbTV application keys included in the keyset of the regular HbbTV® application.
- If anything other than a regular HbbTV® application has input focus, the operator application shall have access to regular HbbTV application keys and operator application keys if these are not taken by the application or user interface that has input focus.

Concerning the key events defined as 'Only available to applications once activated' in clause 10.2.2.1 of ETSI TS 102 796 [], an operator application shall be activated immediately when launched.

Subject to the bilateral agreement, a running operator application shall receive key events for the operator application keys listed in table 17 that it has requested to receive using the KeySet API, irrespective of the state that the operator application is in and regardless of whether it has input focus.

Requests by a regular HbbTV® application to receive key events for the operator application keys or the system keys listed in table 17 shall be refused and those key events shall not be delivered to a regular HbbTV® application. Requests by a privileged operator application to receive key events for the system keys listed in table 17 shall be refused and those key events shall not be delivered to a privileged operator application.

Operator-specific operator applications shall be able to register system keys via the KeySet API. The full set of system keys available to an operator-specific operator application is subject to the bilateral agreement and may include keys not listed in the present document. If an operator-specific operator application has successfully requested some system keys, those keys shall be sent to the operator application if not acted on by the terminal irrespective of the state that the operator application is in. For example, no key shall be sent if either the terminal has acted on a press of the TEXT or TXT key to implement the "mechanism to start and stop digital teletext applications" from ETSI TS 102 796 [] or if the terminal has acted on a press of the "EXIT or comparable button" to terminate a regular HbbTV®.

Table 17: Key events and their type

Button (for conventional remote controls	DOM-2 Key event	Type
4 colour buttons (red, green, yellow, blue)	VK_RED, VK_GREEN, VK_YELLOW, VK_BLUE	Regular HbbTV application key
4 arrow buttons (up, down, left, right)	VK_UP, VK_DOWN, VK_LEFT, VK_RIGHT	
ENTER or OK button	VK_ENTER	
BACK button	VK_BACK	
Number keys	VK_0 to VK_9 inclusive	
Play, stop, pause	VK_STOP and either VK_PLAY and VK_PAUSE or VK_PLAY_PAUSE	
Fast forward and fast rewind	VK_FAST_FWD, VK_REWIND	
Record	VK_RECORD	
2 program selection buttons (e.g. P+ and P-, or Channel Up and Channel Down)	VK_CHANNEL_UP, VK_CHANNEL_DOWN	Operator application key
INFO, or comparable button	VK_INFO	
GUIDE, or comparable button	VK_GUIDE	
Channel list or comparable button	VK_CHANNELS (see note 2)	
MENU, or comparable button	VK_MENU	
Volume control buttons	VK_VOLUME_UP, VK_VOLUME_DOWN, VK_MUTE	
Component selection buttons	VK_SUBTITLE VK_AUDIO_TRACK (see note 2)	
Audio description	VK_AUDIO_DESC (see note 2)	
TEXT or TXT or comparable button	VK_TELETEXT	System key
EXIT or comparable button	VK_EXIT (see Note 3)	Operator application key
NOTE 1: Mapping the above to remote controls with a reduced number of buttons is outside the scope of the present document and a matter for the bilateral agreement. NOTE 2: This constant is defined for the first time in the present document and not included in the list of virtual key codes in the OIPF Web Standards TV Profile [i.2] or earlier works. These constants shall be available via a KeyEvent interface. For example VK_CHANNELS can be accessed as KeyEvent.VK_CHANNELS. NOTE 3: If an operator application registers to receive key events for this key, the terminal no longer automatically transitions the operator application to the background state from the foreground and transient states when the key is pressed and the operator application has to take responsibility for implementing suitable behaviour for this key (see clause 6.3.3.3). If a linear channel is selected, this should include transitioning to the background state.		

10.1.4 HbbTV® reported capabilities and option strings

The present document does not define any extensions to the reported capabilities and option strings defined in clause 10.2.4 of ETSI TS 102 796 [].

NOTE: Diversity between terminal models should be addressed as part of the bilateral agreement. Private extensions to the XML capabilities are possible.

11 Security

11.1 Overview

From time to time terminals will require privileged access to resources available from specific servers. Whilst it is usual for remote terminals to authenticate servers over TLS, further provision has to be made to enable terminals to be recognized and securely authenticated by servers in a managed way as a basis for permitting such access (known as client authentication).

Similarly, when distributing application packages towards a specific population of terminals and excluding use by terminals not intended to form part of the targeted population, there needs to be a means to ensure the application packages are kept confidential during the distribution process and are accessible only by the intended targets.

The present document utilizes asymmetric public key cryptography to achieve both of the above and requires the terminals to be provisioned with two securely stored confidential private keys (each half of a public-private key pair) to this end. The private key used for client authentication also has an associated certificate, provisioned within the terminal and known as the client certificate, conveying the client authentication public key. The private key used for decrypting distributed application packages is known as the terminal packaging key.

Digital certificates provide a mechanism to associate keys with known entities and are used in the terminal to associate terminal held keys with their manufacturer. Using a public key infrastructure (PKI) allows cooperating parties to manage, use and revoke these digital certificates.

Terminals may use the trusted certificate authorities as defined in clause 11.2.3 of ETSI TS 102 796 [] to provide server-side identity management.

Operators wishing to authenticate the terminal may request the client certificate over TLS.

Operators wishing to target an application package towards a terminal may use the terminal packaging public key as described in the present document.

When verifying the authenticity of an application package received by the terminal, terminals may use the operator's signing certificate to identify the operator that created the operator application package and verify that it has not been tampered with.

Clause 11.2.1 outlines the use and profile of the client certificate. Clause 11.3 outlines the encryption of the operator application package and use of the terminal packaging key.

11.2 Device and Server Authentication

11.2.1 Mutual TLS Authentication

11.2.1.1 Overview

Terminals and operator applications may use mutual TLS authentication to verify the trust of one another. Mutual TLS authentication refers to the server requesting and verifying an X.509 client certificate during the TLS handshake process.

Mutual TLS authentication may be used during the retrieval of the encrypted application package via HTTP, subject to the bilateral agreement. Mutual TLS authentication may be a prerequisite to accessing server side resources as requested by the running operator application.

The client certificate specified here, issued by a manufacturer controlled certificate authority, attests only to the terminal model and vendor as described in 7.3.2.4 "HTTP User-Agent header" of ETSI TS 102 796 []. The manufacturer controlled certification authority shall ensure that this assertion correctly corresponds to the terminal in which the certificate and associated private key are embedded.

Furthermore, the operator relying on this certificate (after successful terminal authentication over TLS) to determine the remote terminal's model and vendor shall not widen the assumptions regarding the terminal characteristics without further trusted information in the context of the bilateral agreement. For example, knowledge of whether this terminal, identified by the model and vendor presented in the client certificate, is an HbbTV® compliant terminal or is capable of running an operator app needs to be determined by other means outside the scope of the present document.

Operators shall use TLS server certificates compatible with the specifications detailed in clause 11.2 "TLS and Root Certificates" of ETSI TS 102 796 [].

11.2.1.2 Client certificate

11.2.1.2.1 Client certificate overview

Terminals shall provide a client certificate if requested to do so during the TLS handshake process.

Client certificates may have a hierarchy depicted in figure 2. The client certificate shall be issued by either by an intermediate certificate authority or a self-signed root certificate authority. It is good practice for the hierarchy to contain at least one intermediate certificate authority.

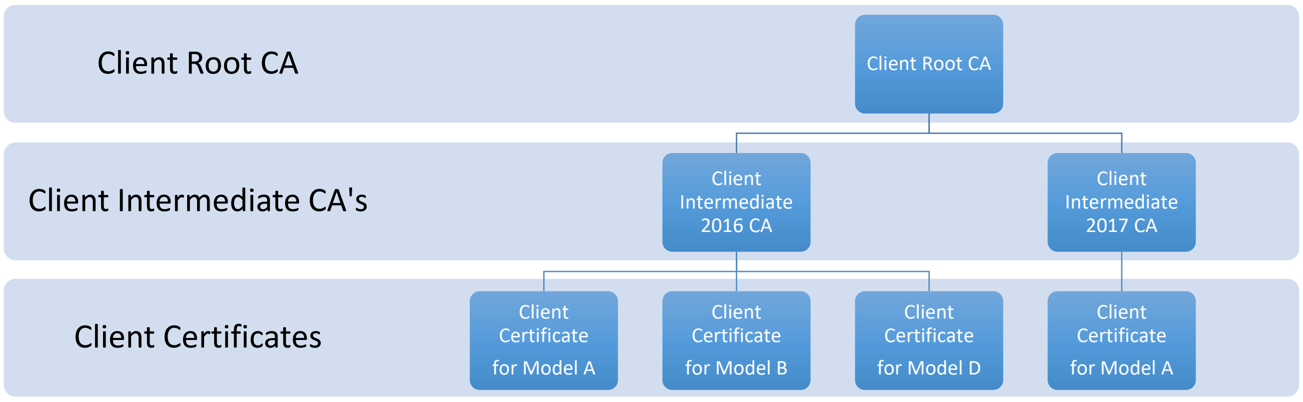
Client Root CA
Client Root CA
Client Intermediate CA's
Client Intermediate 2016 CA
Client Intermediate 2017 CA
Client Certificates
Client Certificate for Model A
Client Certificate for Model B
Client Certificate for Model D
Client Certificate for Model A

Figure 2: Example TLS Client Certificate Hierarchy

NOTE 1: Manufacturers may partition yearly models by attributing the year with a Client Intermediate Certificate Authority. This technically would allow for modifications of certificate specifications in future years. Manufacturers who supply different consumer Brands may partition at a higher level. Some manufacturers may have complex factory locations and thus could be grouped by location as well.

Terminals shall provide certificate chains including the client certificate that are valid for at least 25 years from the point the client certificate was included on any terminal.

NOTE 2: This means that the 25 year validity requirement applies to the first terminal of a particular model. If a manufacturer makes a terminal model for a year without changing the certificate, then the last one off the production line may have certificates valid for just 24 years from date of manufacture.

Negotiation and delivery of the TLS client certificate to the server is defined by the TLS Specification [10].

TLS client certificates shall comply with IETF RFC 5280 [11].

11.2.1.2.2 Operational considerations

In order that operators are able to remotely authenticate the manufacturer terminals over TLS, the manufacturers shall provide and release to operators subject to the bilateral agreement the appropriate Client CA anchor certificate, (either the self-signed Client Root CA certificate or alternative Client Intermediate CA Certificate) in the X.509 representation as described below.

During the operational certificate validation phase of client to server authentication, operators shall reject terminals that contain a client certificate where the chain of trust cannot be verified in the manner of IETF RFC 5280 [11].

Operators should regularly (e.g. on a daily basis) check manufacturer's CRLs distribution points from the trusted Client CA anchor certificate for all revoked certificates. Operators should reject revoked client certificates during the TLS handshake process subject to the bilateral agreement.

11.2.1.2.3 Client Root and Intermediate Certificate Authority Certificate Profiles

The manufacturer's Client Root Certificate Authority (Client Root CA) shall issue either a client certificate or one or more Intermediate Certificate Authority (Client Intermediate CA) certificates. The Client Root CA shall manage a certificate revocation list dedicated to the revocation of these intermediate CA certificates. The Client Intermediate CA shall manage a certificate revocation list dedicated to their respectively issued certificates according to the CA hierarchy.

Manufacturers shall use either the Client Root CA certificate or a Client Intermediate CA certificate as the Client CA's trust anchor certificate. Manufacturers shall provide the Client CA trust anchor certificate to operators subject to the bilateral agreement.

- The Client Root CA certificate shall be a standard X.509 v3 certificate, conforming to IETF RFC 5280 [11] with the additional constraints profiled in table 18 and table 19.
- The public key shall be an RSA key of length at least 2 048 bits.
- The Client Root CA certificate subject should not include any personal email addresses.
- Providers of Client Root CA certificate shall provide a Certificate Revocation List (CRL) service.
- The Client Root CA certificates shall be self-signed.

Intermediate CA certificates shall conform to IETF RFC 5280 [11] with the additional constraints detailed in table 18 and table 19.

Table 18: Client Root and Intermediate Certificate Authority Certificate Profiles - Basic Fields

Field Name	Description
`signatureAlgorithm`	The value of the `signatureAlgorithm` field shall be set to either sha256WithRSAEncryption or sha384WithRSAEncryption, as defined in IETF RFC 4055 [13].
`signature`	The value of the `signature` field shall be set to the same value as the `signatureAlgorithm` field.

Table 19: Client Root and Intermediate Certificate Authority Certificate Profiles - Extended Fields

Field Name	Critical	Description
`authorityKeyIdentifier`	FALSE	May be omitted on the self-signed Client Root CA certificate, otherwise shall be present.
`subjectKeyIdentifier`	FALSE	Shall be present on both the Client Root CA certificate and any Client Intermediate CA certificates.
`keyUsage`	TRUE	Shall be present and the bit fields for keyCertSign and cRLSign shall be set.
`basicConstraints`	TRUE	Shall be present and cA field shall be set to true.
`cRLDistributionPoints`	FALSE	Shall be present and include an http url represented as a fullName of type uniformResourceIdentifier.

11.2.1.2.4 Client certificate profile

The terminal shall contain a valid client certificate and associated private key to enable the terminal to authenticate to a server over TLS. The client certificate shall be a standard X.509 v3 certificate conforming to IETF RFC 5280 [11] with the additional constraints profiled in table 20 and table 21.

- The public key shall be an RSA key of length at least 2 048 bits.
- The client certificate subject should not include any personal email addresses.
- The client certificate shall be signed by either the Client Root CA or a Client Intermediate CA.

Table 20: Client Certificate Profile - Basic Fields

Field Name	Description
`signatureAlgorithm`	The value of the `signatureAlgorithm` field shall be set to either sha256WithRSAEncryption or sha384WithRSAEncryption, as defined in IETF RFC 4055 [13].
`signature`	The value of the `signature` field shall be set to the same value as the `signatureAlgorithm` field.

Field Name	Description
`subject`	The value of the Organization ('O=') attribute of the `subject` field shall be set to the value of the <vendorName> field of the HTTP User-Agent header as defined in clause 7.3.2.4 of ETSI TS 102 796 []. The value of the CommonName ('CN=') attribute of the `subject` field shall be set to either the <modelName> or the <familyName> of the HTTP User-Agent header as defined in clause 7.2.3.4 of ETSI TS 102 796 []. Any other mandatory attributes of the `subject` field defined in IETF RFC 5280 [11] shall be included in the certificate without further constraints. These attributes may be ignored by the operator.

Table 21: Client Certificate Profile - Extended Fields

Field Name	Critical	Description
`authorityKeyIdentifier`	FALSE	The `authorityKeyIdentifier` field shall be present.
`keyUsage`	TRUE	Shall be set to digitalSignature for this certificate type.
`extKeyUsage`	TRUE	Shall be set to id-kp-clientAuth for mutual TLS Authentication.
`basicConstraints`	TRUE	Shall be present and cA field shall be set to false.
`cRLDistributionPoints`	FALSE	Shall be present and include an HTTP URL represented as a fullName of type uniformResourceIdentifier.

11.2.2 Device authentication in broadcast (informative)

There is no equivalent of explicit device authentication using TLS client certificates for broadcast-only deployments. In order for an operator application to be installed, it needs to be decrypted using the private key corresponding to the terminal packaging certificate(s) of a trusted manufacturer. It is recommended that, if an operator application runs only on some models from a manufacturer, then the terminal packaging certificate(s) used on the models where that operator application can run should not be used on other models. This could be done using one terminal packaging certificate per model or one terminal packaging certificate for that specific set of models, e.g. for one or more specific product families. If an operator application needs to distinguish between individual trusted models, LocalSystem.modelName and similar properties can be used.

11.3 Operator application authentication

11.3.1 Encrypted application package overview

This clause defines how to create the encrypted application package using the application ZIP file as defined in clause 7.4 with authentication information. The file is signed, encrypted and packaged into a Cryptographic Message Syntax (CMS) message (IETF RFC 5652 [12]) for delivery to a terminal as defined in clause 11.3.4.

11.3.2 Operator Signing Certificate

In order to sign an application package, an Operator signing key is used and published in an Operator Signing Certificate. Each Operator Signing Certificate is issued by a self-signed Operator Signing Root CA as described in figure 3. Operators may use an intermediate CA to provide better separation between different signing certificates. The example in figure 3 shows an operator's hierarchy with two country-specific Intermediate CA and three signing certificates.

The Operator Signing Root CA certificate shall have the same profile as a Client Root CA certificate, which is defined in clause 11.2.1.2.3. An Operator Signing Root CA certificate and any intermediate certificate installed in the terminal as Operator Signing trust anchor shall have a validity period of at least 25 years from the point of signing the bilateral agreement.

Operators shall release the Operator Signing Root CA certificate and intervening Operator Signing Intermediate CA's certificates to manufacturers under the bilateral agreement. The details around the release of the operator certificate used for operator application authentication as well as further content of the bilateral agreements are out of scope of the present document. Manufacturers shall include this CA in the list of supported operators in those terminals to which the bilateral agreement applies (see clause 6.6.1).

NOTE: Manufacturers may consider a mechanism for adding additional Operator Signing Root CA certificates to their products if they wish to be able to support operator applications that become available after the manufacture of a terminal.

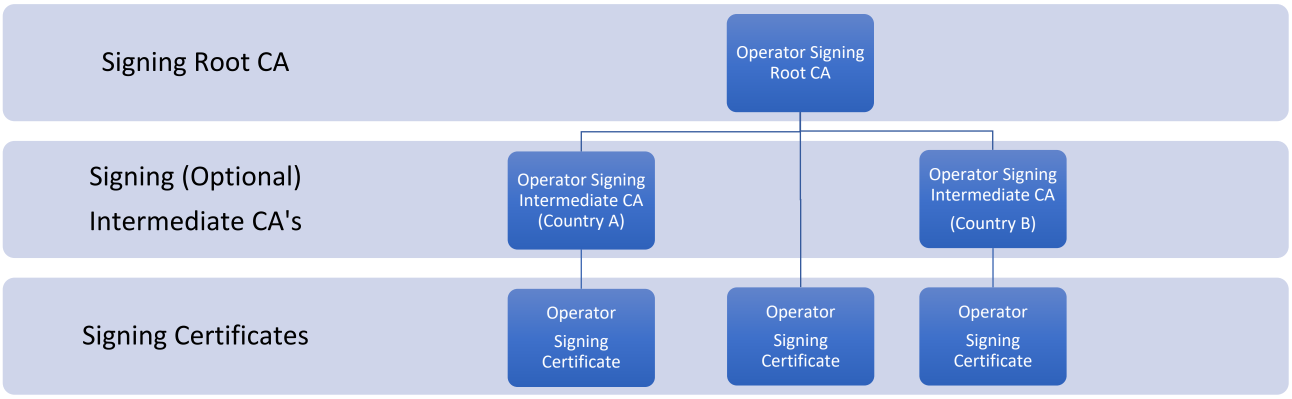
Signing Root CA
Operator Signing Root CA
Signing (Optional) Intermediate CA's
Operator Signing Intermediate CA (Country A)
Operator Signing Intermediate CA (Country B)
Signing Certificates
Operator Signing Certificate
Operator Signing Certificate
Operator Signing Certificate

Figure 3: Example of an Operator Signing Root CA Hierarchy

The Operator Signing Certificate shall be a standard X.509 v3 certificate conforming to IETF RFC 5280 [11] with the following additional constraints:

- The public key shall be an RSA key of length at least 2 048 bits.
- The Operator Signing Certificate subject field should not include any personal email addresses.
- The Operator Signing Certificate shall be signed by either the Operator Signing Root CA or an Operator Signing Intermediate CA.
- The Basic Fields of the Operator Signing Certificate shall further adhere to the constraints in table 22.
- The Extension Fields of the Operator Signing Certificate shall further adhere to the constraints listed in table 23.

The present document only permits key encryption using RSAES-PKCS-v1.5/PKCS#1 v1.5 identified as 'rsaEncryption' in IETF RFC 3447 [18].

Table 22: Operator Signing Certificate Profile - Basic Fields

Field Name	Description
`signatureAlgorithm`	The value of the `signatureAlgorithm` field shall be set to either sha256WithRSAEncryption or sha384WithRSAEncryption, as defined in IETF RFC 4055 [13].
`Signature`	The value of the `signature` field shall be set to the same value as the `signatureAlgorithm` field.
`Subject`	The value of the Organization ('O=') attribute of the `subject` field shall be set to the name of the operator. The value of the CommonName ('CN=') attribute of the `subject` field shall be set to the organisation_id of the operator. The organisation_id shall be encoded as an unsigned decimal integer without any leading zeros. Any other mandatory attributes of the `subject` field defined in IETF RFC 5280 [11] shall be included in the certificate without further constraints.

Table 23: Operator Signing Certificate Profile - Extension Fields

Field Name	Critical	Description
`authorityKeyIdentifier`	FALSE	The `authorityKeyIdentifier` field shall be present.
`keyUsage`	TRUE	The field shall be set to the `digitalSignature` value of this certificate type
`basicConstraints`	TRUE	The `basicConstraints` field shall be present, with `cA` set to `FALSE`.
`cRLDistributionPoints`	FALSE	The `cRLDistributionPoints` field shall be present and include an http url represented as a `fullName` of type `uniformResourceIdentifier`.

11.3.3 Terminal Packaging Certificate

Manufacturers may create a hierarchy as described in figure 4 including the use of Intermediate CAs to provide better separation between different packaging certificates.

Manufacturers shall create a Terminal Packaging Certificate that is either self-signed or signed by a self-signed Manufacturer Packaging Root CA certificate or signed by a Terminal Packaging Intermediate CA. Manufacturers shall use the profiles detailed in clause 11.2.1.2.3 to generate a Manufacturer Packaging Root CA. The Manufacturer Packaging Root CA certificate is not required to have a minimum validity period.

Manufacturers shall release the relevant Terminal Packaging Certificates to operators under the bilateral agreement. The release process is out of scope for the present document. Operators shall use all of the provided Terminal Packaging Certificates in the process of creating the encrypted application package as defined in clause 11.3.4.3.

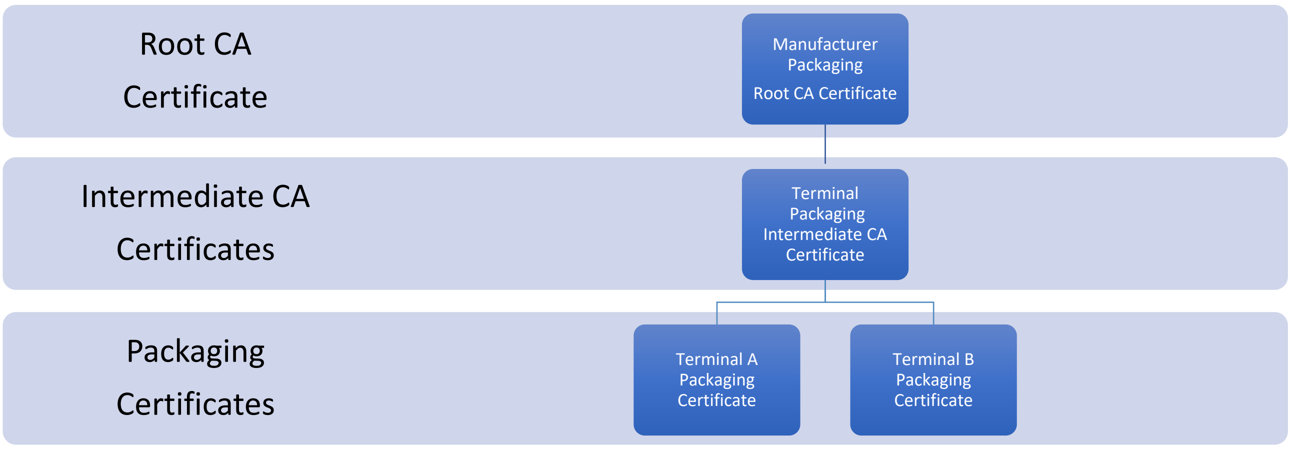
Root CA Certificate
Manufacturer Packaging Root CA Certificate
Intermediate CA Certificates
Terminal Packaging Intermediate CA Certificate
Packaging Certificates
Terminal A Packaging Certificate
Terminal B Packaging Certificate

Figure 4: Example of a Manufacturer Packaging Certificate Hierarchy

The Terminal Packaging Certificate shall be a standard X.509 v3 certificate conforming to IETF RFC 5280 [11] with the following additional constraints:

- The public key shall be an RSA key of length at least 2 048 bits.
- The Terminal Packaging Certificate `subject` field should not include any personal email addresses.
- The Terminal Packaging Certificate shall not expire during the lifetime of the bilateral agreement.
- The Basic Fields of the Terminal Packaging Certificate shall further adhere to the constraints in table 24.
- The Extension Fields of the Terminal Packaging Certificate shall further adhere to the constraints listed in table 25.

The present document only permits key encryption using RSAES-PKCS-v1.5/PKCS#1 v1.5 identified as 'rsaEncryption' in IETF RFC 3447 [18].

Table 24: Terminal Packaging Certificate Profile - Basic Fields

Field Name	Description
`signatureAlgorithm`	The value of the `signatureAlgorithm` field shall be set to either sha256WithRSAEncryption or sha384WithRSAEncryption, as defined in IETF RFC 4055 [13].
`signature`	The value of the `signature` field shall be set to the same value as the `signatureAlgorithm` field.
`subject`	The value of the Organization ('O=') attribute of the `subject` field shall be set to the value of the <vendorName> field of the HTTP User-Agent header as defined in clause 7.2.3.4 of ETSI TS 102 796 []. The value of the CommonName ('CN=') attribute of the `subject` field shall be set to either the <modelName> or the <familyName> of the HTTP User-Agent header as defined in clause 7.2.3.4 of ETSI TS 102 796 []. Any other mandatory attributes of the `subject` field defined in IETF RFC 5280 [11] shall be included in the certificate without further constraints.

Table 25: Terminal Packaging Certificate Profile - Extended Fields

Field Name	Critical	Description
`authorityKeyIdentifier`	`FALSE`	The `authorityKeyIdentifier` field shall be present.
`keyUsage`	`TRUE`	The field shall be set to the `keyEncipherment` value of this certificate type.
`basicConstraints`	`TRUE`	The `basicConstraints` field shall be present, with `cA` set to `FALSE`.
`cRLDistributionPoints`	`FALSE`	The `cRLDistributionPoints` field shall be present and include an http url represented as a `fullName` of type `uniformResourceIdentifier`.

11.3.4 Encrypted application packaging process

11.3.4.1 Encrypted application packaging process overview

Clauses 11.3.4.2 and 11.3.4.3 describe the signing and encryption processes used by the operator to generate an encrypted application package, using the Operator Signing Certificate (see clause 11.3.2) and Terminal Packaging Certificates (see clause 11.3.3).

Clauses 11.3.4.4 and 11.3.4.5 describe the decryption and signature verification processes performed on the terminal after download of the encrypted application package.

11.3.4.2 Operator application signing process

This clause describes the process of packaging an application ZIP file into a CMS `SignedData` structure. This structure is then encrypted into a CMS `EnvelopedData` structure according to clause 11.3.4.3.

A CMS `SignedData` shall be constructed according to section 5 of IETF RFC 5652 [12], with the following additional requirements:

- The signer-specific message-digest algorithm for generating the `messageDigest` shall be either SHA256 or SHA384 (as signalled through in the Operator Signing Certificate).
- Although IETF RFC 5652 [12] specifies that certificates are optional, the Operator Signing Certificate and any Operator Intermediate CAs shall be included in the `certificates` block within the CMS `SignedData`. The Operator Signing Certificate shall be profiled as defined in clause 11.3.2.
- The application ZIP file shall be included in the CMS `SignedData` under the `EncapsulatedContentInfo` field, with the `eContentType` field set to `id-data`.

The following provides an informative OpenSSL example for generating a CMS `SignedData` using SHA256 and an Operator Signing Certificate and corresponding private key.

OpenSSL 1.0.2j Example:

```
# openssl cms -sign -in OpApp.zip -binary -nodetach -nosmimecap -inkey Operator-Private.key -signer
Operator-Certificate.crt -outform DER -out OpApp-zip-signed.cms -md sha256
```

where:

cms	uses the CMS utility
-sign	sign using the supplied Operator Signing Certificate and private key
-binary	the input message is handled as binary data
-nodetach	the data being signed is included in the `EncapsulatedContentInfo`
-nosmimecap	exclude the list of supported algorithms from the signed attributes
-inkey	the private key that corresponds to the Operator Signing Certificate
-signer	the Operator Signing Certificate
-outform	sets the CMS output format to DER encoded structures
-out	the output filename of the CMS `SignedData`
-md	the digest algorithm to use when signing

11.3.4.3 Process for encrypting an application package

To create the final encrypted application package, the operator shall use the CMS `SignedData` defined in clause 11.3.4.2 to create an CMS `EnvelopedData` defined in section 6 of IETF RFC 5652 [12], with the following additional requirements:

- The `contentEncryptionAlgorithm` used to encrypt the data shall be set to either `aes-128-cbc` or `aes-256-cbc`.
- In the creation of the CMS `EnvelopedData`, Operators shall use the Terminal Packaging Certificates of all potential recipients of the encrypted application package. The encrypted content-encryption key and other recipient-specific information shall be used as defined in Section 6.2 of IETF RFC 5652 [12] using a ktri field of type KeyTransRecipientInfo as described in section 6.2.1 of IETF RFC 5652 [12].
- Operators shall ensure that the content-encryption key is randomly generated for each generation of the encrypted applicationPackage as detailed in section 6.2 of the IETF RFC 5652 [12].

The following provides an informative OpenSSL example for encrypting the CMS `SignedData` using aes-128-cbc cipher algorithm into an encrypted application package. The example provides both manufacturer 1 with Terminal Packaging Certificates A & B and manufacturer 2 with a Terminal Packaging Certificate C the ability to decrypt the encrypted application package.

OpenSSL 1.0.2j Example:

```
# openssl cms -encrypt -aes-128-cbc -binary -in OpApp-zip-signed.cms -out OpApp-zip-encrypted.cms -
outform DER manufacturer1-terminal-packaging-cert-A.crt manufacturer1-terminal-packaging-cert-B.crt
manufacturer2-terminal-packaging-cert-C.crt
```

where:

cms	uses the CMS utility
-encrypt	encrypts the input CMS `SignedData`
-aes-128-cbc	the encryption algorithm/cipher used
-binary	the input message is handled as binary data

-in	the input filename of the CMS `SignedData`
-outform	sets the CMS encrypted output message to DER encoded structures
-out	the output name of the encrypted application package
certificates	one or more public certificates that may decrypt the package

11.3.4.4 Process for decrypting an application package

After downloading the encrypted application package as defined in clause 6.1.7, terminals shall decrypt the encrypted application package to access the application zip file according to the following process.

The terminal shall use the private key of the Terminal Packaging Certificate to decrypt the `EncryptedKey` of the relevant `RecipientInfo` field of the received encrypted application package as defined in section 6.2 of IETF RFC 5652 [12]. The terminal shall use the decrypted key to decrypt the `encryptedContent` field, resulting in the CMS `SignedData` structure.

The terminal shall support the following cipher algorithms:

- aes-128-cbc
- aes-256-cbc

Terminals encountering other cipher algorithms shall fail the decryption process and follow the process outlined in clause 6.1.9.

The following provides an informative OpenSSL example for decrypting an encrypted application package using the Terminal Packaging Certificate private key and certificate.

OpenSSL 1.0.2j Example:

```
# openssl cms -decrypt -binary -inform DER -in OpApp-zip-encrypted.cms -keyform PEM -inkey
manufacturer1-terminal-packaging-cert-A.key -recip manufacturer1-terminal-packaging-cert-A.crt -out
OpApp-zip-decrypted.cms
```

where:

cms	uses the CMS utility
-decrypt	decrypts the input encrypted application package
-binary	the message is handled as binary data
-inform	the input message is a DER encoded format
-in	the filename of the encrypted application package
-keyform	the format of the private key file
-inkey	the Terminal Packaging Certificate associated private key that will decrypt the encrypted symmetric key
-recip	sets the Terminal Packaging Certificate that shall match in the encrypted package
-out	the output name of the CMS `SignedData`

11.3.4.5 Application ZIP package signature verification process

After decrypting the encrypted application package as defined in clause 11.3.4.4, terminals shall verify the resulting CMS `SignedData` according to the following process.

Terminals shall use the Operator Signing Root CA to verify the certificates included in the `certificates` block of the CMS `SignedData` structure as detailed in section 5.1 of IETF RFC 5652 [12].

Terminals shall extract the application ZIP file from the `encapContentInfo` block of the CMS `SignedData`. Terminals shall fail and reject the verification if any of the following conditions occur:

- The certificate chain fails certificate path validation as defined in clause 6 of RFC 5280 [11] (this includes a check that none of the certificates have expired). The required check that certificates have not been revoked shall be performed by obtaining the appropriate CRLs using the cRLDistributionPoints extension (see table 23).
- The Operator Name, as signalled via the Organization ('O=') attribute of the `subject` field, or the organisation_id, as signalled via the CommonName ('CN=') attribute of the `subject` field do not match those defined in the bilateral agreement for the operator whose organisation_id is found during the discovery process in clause 6.1.5.
- The value of the `message-digest` field contained in the CMS `SignedData` structure does not match with the terminal generating a message-digest of the extracted application ZIP file when applying the hashing function communicated via the `SignatureAlgorithm` field.

If verification fails, the terminal shall follow the process outlined in clause 6.1.9.

The following provides an informative example where the decrypted application ZIP file is verified with the Operator Signing Root CA. The example only covers validating the operator's certificate chain and the message-digest of the application ZIP file. It does not include checking certificates for revocation using CRLs.

OpenSSL v1.0.2j Example:

```
openssl cms -verify -binary -in OpApp-zip-decrypted.cms -inform DER -out OpApp.zip -CAFile Operator-
Certificate-Authority.crt
```

where:

cms	uses the CMS utility
-verify	verifies the input message
-binary	the message is handled as binary data
-in	the input CMS `SignedData`
-inform	the input message is a DER encoded format
-out	the output name of the verified application ZIP file
-CAFile	the Operator Root Certificate Authority

11.4 CI Plus

11.4.1 CI Plus communication

Clause 4.2.3.4.1.1 of OIPF CSP [15] specifies how and when the terminal sets up a session to the CI Plus SAS resource (as specified in CI Plus Specification v1.3.2 [5]) for messaging between the terminal and the CICAM. In addition to this session and at the same time, the terminal shall open an additional session to the SAS resource by sending a SAS_connect_rqst() APDU to the CICAM with the private_host_application_ID value of 0x68626274766f7061. If the CICAM accepts this connection, the terminal shall use this additional session for all messages between the operator application and the CICAM and the session defined by OIPF for all messages between the regular HbbTV® application and the CICAM. If the CICAM refuses this connection, the terminal shall use the same session for all messages relating to both applications.

11.4.2 IP Delivery CICAM Player Mode

Terminals supporting the "IP delivery CICAM player mode" as defined in the DVB Extensions to CI Plus ETSI TS 103 205 [7] shall support the use of "Host-initiated playback" (as defined in clause 8.3.2 of [7]) in combination with the HTML5 video element as defined in Annex F of the present document.

12 Privacy

In the case of privileged operator applications, use of an operator application is entirely a user choice. Terminals are also required to allow the user to uninstall a privileged operator application (see clause 6.7).

In the case of operator-specific operator applications, the operator application essentially forms a part of the terminal. In most cases, the user will have bought or rented the terminal specifically in order to get that operator's services. Operator-specific operator applications do not gain any capabilities that may impact a viewer`s privacy beyond what a terminal manufacturer may do in the software that controls a terminal that does not use operator applications.

Operator applications will need to include appropriate terms and conditions as part of the package and obtain user agreement to these when they are first launched after being installed.

- For a privileged operator application, if the user does not accept the terms and conditions then the user can uninstall the operator application or the operator application can uninstall itself (see clause 6.7).
- For an operator-specific operator application, if the user does not accept the terms and conditions then they have to return the terminal to where they got it from.

The present document does not place any requirements on operator applications that require the sharing of any user data, preferences or behaviour information with the operator.

Operator applications are securely delivered to the terminal and have a capability for secure communications allowing any necessary data exchanges with operators to be kept confidential.

Although the present document uses client certificates as part of authenticating a terminal to an operator, these are used to identify models or product families and not to identify individual devices.

EXAMPLE: A client certificate may be common to all of a manufacturer`s models in a calendar year using a particular hardware and software architecture regardless of physical attributes such as panel size and bezel.

No new identifiers are introduced for use by operator applications. Only those capabilities in ETSI TS 102 796 [] are provided, along with the privacy controls defined for them.

A private persistent storage shall be provided by the terminal for use by operator applications subject to a bilaterial agreement. Annex A.2.2.2 provides further details on the private storage class. The operator application shall comply with all regulatory requirements regarding privacy policy and protecting end-users.

Consumer electronics products such as television sets and set-top boxes normally include the ability for the user to reset the product to the condition it was in when first obtained by the user. This will erase any data stored locally by an operator application.

13 Media synchronization

Operator applications have access to the same media synchronization functions as regular HbbTV® applications. Operator applications shall be able to enable multi-stream and inter-device synchronization for content being presented by an operator application video/broadcast object, A/V control object or HTML5 video element, including for recordings being played back under the control of the operator application.

NOTE: An operator application cannot enable inter-device synchronization for content being presented directly by the terminal or by a regular HbbTV® application, even if the operator application is using a BroadcastSupervisor object.

14 Companion screens

Privileged and operator-specific operator applications shall be able to use application to application communications as defined in clause 14.5 of ETSI TS 102 796 [] simultaneously with a regular HbbTV® application also using that feature. Terminals shall support at least the number of concurrent WebSocket connections defined in clause 14.5.3 of ETSI TS 102 796 [] for an operator application and the same again for a regular HbbTV® application.

Operator applications using application to application communication need to follow the guidance given in clause 14.5.4 of ETSI TS 102 796 [] to avoid collisions with regular HbbTV® applications also using this feature.

The server for application-to-application communications shall be able to accept connections once an operator application has called the getApp2AppLocalBaseURL() method and until the application exits even if no regular HbbTV application is running or if a regular HbbTV application is running but has not called that method.

15 Communication with regular HbbTV® applications

15.1 General

Terminals shall implement a mechanism for communication between operator-applications and regular HbbTV® applications. The implementation and behaviour of this mechanism shall be identical to that of application-to-application communications defined in clause 14.5 of ETSI TS 102 796 [] but with the differences as described in the present document.

The terminal shall implement two endpoints as described in clause 15.2, consisting of an endpoint for operator applications and an endpoint for regular HbbTV® applications.

The "operator application endpoint" shall be able to accept connections from an operator application once it has called the `getOpApp2AppBaseURL()` method and until the operator application exits even if no regular HbbTV® application is running or if a regular HbbTV® application is running but has not called the `getApp2OpAppBaseURL()` method.

Similarly, the "regular application endpoint" shall be able to accept connections from a regular HbbTV® application once it has called the `getApp2OpAppBaseURL()` method and until the regular HbbTV application exits even if no operator application is running or if an operator application is running but has not called the `getOpApp2AppBaseURL()` method.

The terminal shall handle connection requests as described in clause 15.3 and apply the pairing process described in clause 15.4. It shall act as a relay for paired connections as described in clause 15.5.

15.2 Service endpoints provided by the terminal

The terminal shall provide two service endpoints as defined in clause 14.5.2 of ETSI TS 102 796. However, instead of there being a remote endpoint and a local endpoint there shall instead be two local endpoints as follows:

- A local "operator application endpoint" for connecting to by clients that are HbbTV® operator applications.
- A local "regular application endpoint" for connecting to by clients that are regular HbbTV® applications.

Both endpoints shall be local endpoints and therefore a terminal shall not make it possible to connect to either endpoint from other devices within the home network.

The regular application endpoint URL shall be determined by regular HbbTV® applications by the method call `getApp2OpAppBaseURL()` as defined in annex A.2.2.2. This is instead of the JavaScript API defined in clause 8.2.6 of ETSI 102 796.

The operator application endpoint URL shall be determined by operator HbbTV® applications by the method call `getOpApp2AppBaseURL()` as defined in annex A.2.2.2. This is instead of the JavaScript API defined in clause 8.2.6 of ETSI 102 796.

Both endpoints shall satisfy the security requirements of clause 11.7 of ETSI 102 796, where the lifetime of an HbbTV application refers to:

- the lifetime of a regular HbbTV application for endpoint URLs obtained by regular HbbTV applications; and
- the lifetime of an operator application for endpoint URLs obtained by operator applications.

The mixed content requirements that permit connections to WebSocket endpoints defined in ETSI 102 796 clause A.3.13 shall also apply to:

- regular HbbTV® applications when connecting to the "regular application endpoint" regardless of how that application is delivered; and
- operator applications when connecting to the "operator application endpoint" regardless of how that application is delivered.

NOTE: Making the endpoint URLs ***potentially trustworthy*** (in the terminology of [i.9]) can meet this mixed content requirement. From [i.9] section 3.1 and 5.2, it is shown that the origin localhost is ***potentially trustworthy*** if the name resolution rules laid out in [i.10] are followed.

15.3 Handling of new connections from clients

The terminal shall be able to handle new connection requests as specified in clause 14.5.3 of ETSI T 102 796 and shall support the same minimum number of concurrent connections.

15.4 Connection Pairing

Connection pairing shall follow the same process as defined in clause 14.5.4 of ETSI TS 102 796. Connections to the operator application endpoint shall only be paired with connections to the regular application endpoint and vis-versa.

NOTE: The URLs for the endpoints are not advertised via the terminal and service endpoint discovery mechanisms defined in clause 14.7 of ETSI TS 102 796.

15.5 Paired Connections

Paired connections shall behave as defined in clause 14.5.5 of ETSI TS 102 796 including requirements for minimum payload sizes and traffic rates. However, instead of sending the UTF-8 encoded text message `'pairingcompleted'`, the terminal shall send a UTF-8 encoded text message containing a JSON object with the following properties:

- The `status` property shall have the value `'pairingcompleted'`
- The `origin` property shall have the value of the `Origin` header from the WebSocket connection request of the connection that this connection has been paired with:
  - The value sent to the operator application shall be the value of the `Origin` header from the WebSocket connection made by the regular HbbTV® application.
  - The value sent to the regular HbbTV® application shall be the value of the `Origin` header of the WebSocket connection made by the operator HbbTV® application

Here is an example message:

```
{ "status": "pairingcompleted", "origin": "https://tv-apps.example.com/" }
```

Annex A (normative): OIPF specification profile

A.1 Detailed section-by-section definition for volume 5

Table A.1 defines clauses of the OIPF DAE specification [2] that are required by the present document but which are either not required at all by ETSI TS 102 796 [] or where the present document introduces additional requirements beyond what is required by the OIPF DAE specification [2]. Where a class or object is partly required by ETSI TS 102 796 [], the properties and/or methods and/or events required by [] are required by the present document. Only additional requirements are listed here. Methods properties and events that are not required by ETSI TS 102 796 [] and not required by the present document should not be supported unless required by another specification.

Table A.1: Changes to Section-by-section profile of the OIPF DAE specification defined by the HbbTV® specification []

Section, sub-section	Reference in DAE [2]	Status in HbbTV®	Terminals supporting privileged operator applications	Terminals supporting operator-specific operator applications
ObjectFactory API	7.1	M(*)		The `oipfObjectFactory` shall be extended to support a `createProprietaryFunctionProvider` method with no arguments that shall return an instance of the `ProprietaryFunctionProvider` class.
The application/oipfApplicationManager embedded object	7.2.1	M(*)	The `onApplicationLoaded` property and the corresponding `ApplicationLoaded` event shall be called/generated in an operator application when a broadcast-independent regular HbbTV® application started by the operator application with `createChild` as true has successfully loaded as described in section 7.2.1.2 of OIPF DAE specification [2]. The `onApplicationUnloaded` property and the corresponding `ApplicationUnloaded` event shall be called/generated in an operator application when a regular HbbTV® application started by the operator application with *createChild* as true is about to exit.	
The application class	7.2.2	M(*)	The extensions defined in clause A.2.2 shall be supported. Operator applications shall have the same behaviour for the `show`() and `hide`() methods as a regular HbbTV® broadcast-independent application. `show`() and `hide`() should not be called by operator applications. If called, they have no effect on the state of the Operator Application as defined in clause 6.3.3. There is no requirement for terminals to support calling any methods on Application objects that represent different applications from the caller.	
The Keyset class	7.2.5	M(*)	The `otherKeys` and `maximumOtherKeys` properties that are defined to be 'not included' by ETSI TS 102 796 [] shall be supported for operator applications. Only the `getKeyLabel` method remains not included.	
The application/oipfConfiguration embedded object	7.3.1	M(*)	The `localSystem` property is mandatory.	

Section, sub-section	Reference in DAE [2]	Status in HbbTV®	Terminals supporting privileged operator applications	Terminals supporting operator-specific operator applications
The Configuration class	7.3.2	M(*)	Support for read and write access to the following properties is mandatory: • `preferredAudioLanguage` • `preferredSubtitleLanguage` Support for read and write access to the following properties defined in ETSI TS 102 796 [] is mandatory: • `audioDescriptionEnabled` • `subtitlesEnabled` The `setQueryOrganisations` method (see clause A.2.1) shall be supported for operator applications. The `runningOperatorApplication` property (see clause A.2.1) shall be supported for regular HbbTV® applications. The extensions defined in clause A.2.1 are mandatory. NOTE: Read-write access is not extended to the `preferredAudioLanguage47` and `preferredSubtitleLanguage47` properties which remain read only. However, those properties will reflect changes made to the `preferredAudioLanguage` and `preferredSubtitleLanguage` properties.	The requirements for privileged operator applications apply. Support for read and write access to the following properties is mandatory: • `preferredUILanguage` `countryId` NOTE: Read-write access is not extended to the `preferredUILanguage47` property defined in clause A.2.1.2 which is read only. However, that property will reflect changes made to the `preferredUILanguage` property.

Section, sub-section	Reference in DAE [2]	Status in HbbTV®	Terminals supporting privileged operator applications	Terminals supporting operator-specific operator applications
The LocalSystem class	7.3.3	NI	The following properties shall be supported if the terminal supports replacement of the UI elements and related functionality for volume control (see clause 6.4): • volume • mute The following properties and methods are required for terminals supporting ACTIVE_STANDBY and subject to bilateral agreement (see clause 6.3.4): • `powerState` • `timeCurrentPowerState` • `onPowerStateChange`	The requirements for privileged operator applications apply. The following properties are mandatory for terminals supporting operator-specific operator applications: • `vendorName` • `modelName` • `familyName` • `softwareVersion` • `hardwareVersion` • `tuners` The following property is mandatory if the terminal supports the provision of a serial number to operator applications: • `serialNumber`
The TunerCollection class	7.3.8	NI	Not Included	Mandatory
The Tuner class	7.3.9	NI	Not Included	The following properties are mandatory: • `id` • `name` • `idTypes` • `signalInfo` • `frontEndPosition`
The SignalInfo class	7.3.10	NI	Not Included	The following properties are mandatory: • `strength` • `quality`
The application/oipfDrmAgent embedded object	7.6.1	M-C(*), M-M(*), M-P(*)	See clause A.2.6.	The requirements for privileged operator applications apply.
The application/oipfParentalControlManager embedded object	7.9.1	M(*)	The following properties and methods are mandatory: • `isPINEntryLocked` • `setParentalControlStatus` • `getParentalControlStatus` • `unlockWithParentalControlPIN` • `verifyParentalControlPIN` In addition, the extensions and clarifications in clause A.2.8 shall be supported.	The requirements for privileged operator applications apply. The following methods are mandatory: • `setParentalControlPIN`

Section, sub-section	Reference in DAE [2]	Status in HbbTV®	Terminals supporting privileged operator applications	Terminals supporting operator-specific operator applications
The ParentalRatingScheme class	7.9.2	M	Unchanged	Operator-specific operator applications shall have write access to the `threshold` property of the scheme supporting DVB-SI age based rating.
The application/oipfRecordingScheduler embedded object	7.10.1	M-P(*)	As required by ETSI TS 102 796 [] with the modifications in clause A.2.7.	
The ScheduledRecording class	7.10.2	M-P(*)	As required by ETSI TS 102 796 [1]. The following additional properties shall be supported; • `state` • `error` • `scheduled` • `isManual` • `repeatDays`	
Extension to application/oipfRecordingScheduler for control of recordings	7.10.4	M-P(*)	As required by ETSI TS 102 796 [1]. The following additional properties and methods shall be supported with the modifications in clause A.2.7: • `getRecording()` • `onPVREvent()` • `stop()` • `update()`	
The application/oipfRemoteManagement embedded object	7.11.1	NI	Not Included	The following properties and methods are mandatory if the terminal supports operator applications triggering system software update: • `onSoftwareUpdate` • `triggerSoftwareUpdate` • `softwareUpdateStatus` For the triggerSoftwareUpdate method, the token argument is implementation specific and may not be used. When it is not used, the returned integer will never have value 2, invalid token.
video/broadcast embedded object	7.13.1	M(*)	The modifications in clause A.2.4 shall be supported	The modifications in clause A.2.4 shall be supported
Extensions to video/broadcast for recording and timeshift	7.13.2	M(*), M-P(*)	Unchanged	Unchanged
Extensions to video.broadcast for access to EIT p/f	7.13.3	M	The ProgrammesChanged event shall be generated on all occasions when the value of the programmes property changes regardless of the reason for the change.	The ProgrammesChanged event shall be generated on all occasions when the value of the programmes property changes regardless of the reason for the change.

Section, sub-section	Reference in DAE [2]	Status in HbbTV®	Terminals supporting privileged operator applications	Terminals supporting operator-specific operator applications
Extensions to video/broadcast for playback of selected components	7.13.4	M	Unchanged	Unchanged
Extensions to video/broadcast for parental ratings errors	7.13.5	M	Unchanged	Unchanged
Extensions to video/broadcast for DRM rights errors	7.13.6	M-C	Unchanged	Unchanged
Extensions to video/broadcast for current channel information	7.13.7	M	Unchanged	Unchanged
Extensions to video/broadcast for creating Channel lists from SD&S fragments	7.13.8	NI	Unchanged	Unchanged
ChannelConfig class	7.13.9	M(*)	The following methods shall be supported: • `getBroadcastSupervisor` • `createChannelList(String xml)` See clause A.2.3.	The requirements for privileged operator applications apply. The `setChannelList` and `createChannelList` methods (see clause A.2.3) shall be supported. If the terminal makes RF-based channel scans available to operator applications then the properties, methods and events listed below shall be supported for operator-specific operator applications in addition to the `channelList` property required by HbbTV®. • `startScan` • `stopScan` • `createChannelScan ParametersObject` • `createChannelScan OptionsObject` • `onChannelScan` • `onChannelListUpdate` • `ChannelScan event` • `ChannelListUpdate event`
ChannelList class	7.13.10	M(*)	Unchanged	Unchanged
Channel class	7.13.11	M(*)	The `locked` property shall be supported. The `manualBlock` property shall be supported if the terminal makes a UI available for manually blocking channels.	The requirements for privileged operator applications apply. The extensions in clause A.2.9 shall be supported.
Favourite lists	7.13.12, 7.13.13	NI	Unchanged	Unchanged
Extensions to video/broadcast for channel scan	7.13.14	NI	Unchanged	Unchanged

Section, sub-section	Reference in DAE [2]	Status in HbbTV®	Terminals supporting privileged operator applications	Terminals supporting operator-specific operator applications
The ChannelScanEvent class	7.13.15	NI	Unchanged	Unchanged
The ChannelScanOptions class	7.13.16	NI	Unchanged	Mandatory if the terminal makes RF-based channel scans available to operator applications.
The ChannelScanParameters class	7.13.17	NI	Unchanged	Mandatory if the terminal makes RF-based channel scans available to operator applications.
The DVBTChannelScanParameters class	7.13.18	NI	Unchanged	If the terminal makes RF-based channel scans available to operator applications then support for creating instances of these classes is mandatory regardless of what tuners a terminal has.
The DVBSChannelScanParameters class	7.13.19	NI	Unchanged	
The DVBCChannelScanParameters class	7.13.20	NI	Unchanged	
Extensions to video/broadcast for synchronization	7.13.21	NI	Unchanged	Unchanged
The ATSCTChannelScanParameters class	7.13.22	NI	Unchanged	Unchanged

A.2 Modifications, extensions and clarifications to OIPF volume 5

A.2.1 Configuration class

A.2.1.1 Constants

The Configuration class shall be extended with the following constants.

Table A.2: Constants used to define the scope of an operator application

Constant name	Numeric value	Status	Meaning	Related key events
UI_TVMODE	0	M	Used to request suppression of all user interface elements in TV watching mode (with the exceptions defined below): • Especially, this includes info banner, channel banner, channel selection and component selection. • The only exceptions are volume control, parental control, timeshift control, and messages as defined below.	VK_CHANNEL_UP VK_CHANNEL_DOWN VK_INFO VK_CHANNELS VK_AUDIO_TRACK VK_AUDIO_DESC VK_SUBTITLE (See note 1)
UI_VOLUME	1	B	Used to request suppression of any volume change and any UI in regards to volume change and muting	VK_VOLUME_UP VK_VOLUME_DOWN VK_MUTE (See note 1)
UI_PARENTALCONTROL	2	B	Used to request suppression of any UI to unlock a locked operator provided DVB service as detailed in the bilateral agreement. The terminal shall still manage the following: • Automatic locking of DVB services with rated content. • Counting and managing wrong entries of PIN-code.	
UI_TIMESHIFT	3	B	Used to request suppression of any timeshift functionality and any related UI. This is relevant for operator applications offering either local timeshift or network timeshift.	VK_STOP VK_PLAY VK_PAUSE VK_PLAY_PAUSE VK_FAST_FWD VK_REWIND (See note 2)
UI_RECORD	4	B	Used to request suppression of the recording functionality in TV watching mode and any related UI.	VK_RECORD (See note 2)

Constant name	Numeric value	Status	Meaning	Related key events
			This is relevant for operator applications offering either local recording or network recording.	
	5 - 31		*(reserved for future use)*	
UI_MESSAGES_PVR	32	B	Used to request suppression of messages in regards to scheduled recordings (see clause A.2.1.4) and other PVR functions (e.g. broadcast timeshift).	
UI_MESSAGES_REMINDER	33	B	Used to request suppression of scheduled messages (see clause A.2.1.5).	
UI_MESSAGES_DRM	34	B	Used to request suppression of messages initiated by an embedded Conditional Access system or DRM system.	
UI_MESSAGES_SYSTEM	35	B	Used to request suppression of any messages which are not covered by the previous three constants and where the cause of the message is observable by the operator application through HbbTV APIs (e.g. "Signal lost").	
	36 - 63		*(reserved for future use)*	
UI_EPG	64	M	Used to replace the default EPG by the EPG of the operator application.	VK_GUIDE (See note 1)
UI_PVR	65	B	Used to replace the default UI for managing recorded and downloaded content by an equivalent UI of the operator application. If the user has used the manufacturer UI to schedule recordings this may be for channels not offered by the operator (see clause 5.5.1). It may not be appropriate for an operator application to have visibility or manage such recordings.	
UI_HBBTV	66	B-OS	Used to replace the default menu for broadcast-independent HbbTV® applications by an equivalent UI of the operator application.	
UI_MENU	67	B	Used to replace the default main menu by an equivalent UI of the operator application.	VK_MENU (See note 1)
UI_INSTALLATION	68	B-OS	Used to replace the default installation process by the installation process of the operator application.	
	69 - 126		*(reserved for future use)*	
UI_ALL	127	B-OS	Indicates that the terminal shall not provide any UI.	
	128 - 255	B-OS	*(user defined)* *(See note 3)*	

Constant name	Numeric value	Status	Meaning	Related key events
NOTE 1: If the operator application requests suppression of this group of UI elements, the operator application should also request to receive the listed key events of operator application keys if they are supported by the terminal. NOTE 2: If the operator application requests suppression of this group of UI elements, the operator application should also request to receive the listed key events of regular HbbTV application keys. NOTE 3: This range can be used by the manufacturer to define additional and varied terminal-specific UI elements.				

Table A.3: Key to status column

Status	Meaning
M	Support of the request to suppress this group of UI elements is mandatory for both privileged and operator-specific operator applications.
B	Support of the request to suppress this group of UI elements is subject to bilateral agreement for both privileged and operator-specific operator applications.
B-OS	Support of the request to suppress this group of UI elements is subject to bilateral agreement for operator-specific operator applications.

A.2.1.2 Properties

The Configuration class shall be extended with the following property.

`readonly Integer[] runningOperatorApplication`
When read by a regular HbbTV® application, if an operator application is currently running and the HbbTV® application reading the property is permitted to query this (see clause A.2.1.3) then the value of this property shall be an identifier for the running operator application where index 0 in the array contains the organisation_id and index 1 in the array contains the application_id. Otherwise the value of this property shall be null.

`readonly String preferredUILanguage47`
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 language code as defined according to IETF BCP47 - RFC 5646[27]. The indicated set of languages and the order of preference shall be consistent with those found in the `preferredUILanguage` property.

A.2.1.3 Methods

The Configuration class shall be extended with the following methods.

`void setQueryOrganisations( Integer org_ids[] )`	
Description	When called by an operator application, sets the list of organisation_ids from which regular HbbTV® applications are permitted to query if this operator application is running. If called by a regular HbbTV® application then this method shall have no effect. The default is that no organisation_ids are permitted to make this query.
Arguments	An array of organisation_ids as defined in ETSI TS 102 809 [24].

`Integer[] replaceUIElements ( Integer elements[] )`		
Description	Defines which UI elements the operator application wishes to provide instead of the terminal. The method shall return a list of the successfully replaced UI elements. If a running operator application calls this method more than once then the set of requested elements in the most recent call shall supersede the set requested in all previous calls. Specifically, an operator application that no longer wishes to replace a particular UI element just omits this from the set requested.	
Arguments	`elements`	Array of Integer constants taken from table A.2 indicating the UI elements and related functionality of the terminal that are replaced by the operator application.

A.2.1.4 Replacing UI relating to scheduled recordings

A.2.1.4.1 Messages

If the terminal supports replacing of messages relating to scheduled recordings, and if the operator application has requested this replacement (by calling method `replaceUIElements` with argument `UI_MESSAGES_PVR`), the behaviour shall be as defined in table A.4.

Table A.4: Replacing messages in regards to scheduled recordings

Use case	Behaviour
Conflict or other problem is already known at the moment of scheduling a new recording via the operator application.	The terminal shall not show any message in response to the operator application programming a scheduled recording. The `state` property of the `ScheduledRecording` object shall be set to `RECORDING_ERROR`. The `error` property of the `ScheduledRecording` object shall be set according to the error case.
Conflict or other problem is already known at the moment of scheduling a new recording via the terminal or a regular HbbTV® application.	The terminal shall not suppress its UI. (See note)
Scheduled recording is about to start, but due to missing resources the terminal needs to do at least one of the following to start recording: • Change current channel • Stop recording of the current channel • Stop recording of another scheduled recording (e.g. due to live event overrunning or a series recording where the event was unknown at the time of scheduling the other recordings) The scheduled recording has originally been programmed via the operator application or the terminal or a regular HbbTV® application.	The terminal shall not show any message in this case. Between two and five minutes before the scheduled recording is due to start, the terminal shall send a `PVREvent` with its state set to "The recording is due to start in a short time" to the operator application. The `state` property of the `ScheduledRecording` object that has generated the `PVREvent` shall be set to `RECORDING_ERROR`. The `error` property shall be set according to the error case. Once that event has been sent, the terminal should not allow for any rescheduling or new programmings of scheduled recordings that would interfere with the impending recording. Each time that the operator application is started and registers to replace messages relating to scheduled recordings, the terminal shall check for scheduled recordings that are "due to start in a short time" as defined above. The terminal shall then send the `PVREvent` as defined above to any registered listeners.
Requested recording of current channel fails due to missing resources.	Showing a message in this case shall not be controlled by replacing PVR messages (i.e. `UI_MESSAGES_PVR`) but by replacing UI in regards to recording (i.e. `UI_RECORD`).
Requested timeshift fails due to missing resources.	Showing a message in this case shall not be controlled by replacing PVR messages (i.e. `UI_MESSAGES_PVR`) but by replacing UI in regards to timeshift (i.e. `UI_TIMESHIFT`).
Requested channel change fails as there is an ongoing recording on current mux and no additional tuner is available.	Showing a message in this case shall not be controlled by replacing PVR messages (i.e. `UI_MESSAGES_PVR`) but by replacing standard UI in TV watching mode (i.e. `UI_TVMODE`).
NOTE: The bilateral agreement may define that scheduled recordings can only be programmed via the operator application.	

A.2.1.4.2 Conflict resolution (informative)

Details of the design and implementation of the actual conflict resolution mechanism are out of scope of the present document.

The operator application may provide a UI which allows conflict resolution based on the user's choice. The operator application may also implement an automatic algorithm for conflict resolution which applies for instance if the user is absent.

To resolve a conflict, the operator application may use one or several of the following methods:

- Remove a scheduled recording that has not already been started by using method `remove` of the `oipfRecordingScheduler` embedded object.
- Stop a scheduled recording that has already been started by using method `stop` as defined in the extensions to the `oipfRecordingScheduler` embedded object.
- Update a scheduled recording by using method `update` as defined in the extensions to the `oipfRecordingScheduler` embedded object (e.g. if the operator provides information on alternative broadcastings of the same content).
- Stop recording current channel by using method `stopRecording` of the `video/broadcast` embedded object.
- Stop timeshift mode of current channel by using method `stopTimeshift` of the `video/broadcast` embedded object.
- Change current channel.

If the operator application does not resolve the conflict, the terminal may apply its own default algorithm for conflict resolution (without displaying any message or other user interface).

A.2.1.5 Replacing reminders (informative)

Details of the replacement of reminders is out of scope of the present document.

An operator application which is always running could implement management and display of reminders by itself without any additional interface to the terminal using the Web Notifications API (see clause 8.4.1). This may apply to operator-specific operator applications.

If the operator application is not always running, and the terminal provides a user interface for programming and display of reminders, the replacement of scheduled messages by the operator application may require the definition and implementation of proprietary functions as defined in clause 8.3.1. This may include:

- A method which the operator application can use to inform the terminal about a new reminder.
- A listener concept through which the terminal informs a running operator application when a reminder is due.

A.2.2 Application class

A.2.2.1 Properties

The Application class shall be extended with the following properties.

`function onOperatorApplicationStateChange ( String oldState, String newState )`
The function that shall be called immediately prior to the terminal moving the operator application to a new state. The specified function is called with the arguments, `newState`, which identifies the new state and `oldState`, which identifies the old state. The `newState` and `oldState` arguments shall be encoded as defined for the `opAppState` property.

`function onOperatorApplicationStateChangeCompleted ( String oldState, String newState )`
The function that shall be called after the terminal has finished moving the operator application to a new state. The specified function is called with the arguments, `newState`, which identifies the new state and `oldState`, which identifies the old state. The `newState` and `oldState` arguments shall be encoded as defined for the `opAppState` property.

`String opAppState`
When read by an operator application, this property shall return a String identifying which state the application is in. This shall be encoded as follows -`foreground`, `background`, `transient`, `overlaid-foreground` or `overlaid-transient`. If read by a regular HbbTV® application, `undefined` shall be returned.

`function onOpAppUpdate( String updateEvent )`
The function that is called when the operator application's update state is changed. The specified function is called with the following argument: • `String updateEvent` - The event type that caused the invocation of this function. One of:

Value	Description
"SOFTWARE_DISCOVERING"	The terminal is in the process of discovering updates for the operator application software.
"SOFTWARE_DISCOVERY_FAILED"	The process of discovering updates for the operator application software failed.
"SOFTWARE_CURRENT"	The process of discovering updates for the operator application software successfully rediscovered the installed version of the operator application software, and no valid updates were signaled in the (XML) AIT(s). This terminates the software update process.
"SOFTWARE_DOWNLOADING"	New software for the operator application is in the process of being downloaded. This event type may be signalled at multiple times during the download of the new software, indicating positive progress.
"SOFTWARE_DOWNLOAD_FAILED"	The download of new software has failed.
"SOFTWARE_DOWNLOADED"	A new software version of the operator application has been downloaded but has not yet been installed. Applications can now save relevant data that should survive an update.
"SOFTWARE_UNPACKING"	A new version of the operator application has been verified after download and is about to be unpacked as defined in clause 6.1.8.
"SOFTWARE_INSTALLATION_FAILED"	The new version of the operator application was successfully downloaded but could not be installed.

There is no event for a successful update as the operator application will be restarted at that point.

`function onOperatorApplicationContextChange ( String startupLocation, [String launchLocation] )`
The function shall be called due to a request made by the user from the terminal. The specified function is called with the argument `startupLocation`, which identifies the location for the operator application to display. An optional `launchLocation` argument may be provided to indicate where the context event was triggered from. The `launchLocation` argument shall be an appropriate value from table 10 and the `startupLocation` from table 11 or some other value as defined in the bilateral agreement.

A.2.2.2 Methods

The Application class shall be extended with the following methods.

`Boolean opAppRequestTransient()`	
Description	Requests the terminal to move the calling operator application to transient state or requests the terminal to restart the timer started after the previous successful call to this method. The call shall be successful and the request granted if the conditions for a successful call as listed in clause 6.3.3.4 of the present document are met. If the request is granted then: • if the operator application is already in the transient or overlaid transient state, then: – the terminal shall restart the timer with a duration of 1 minute and – the method shall return `true` • else – the terminal shall start a timer with a duration of 1 minute and – the method shall return `true`. – the Application shall be notified via an event dispatched to the `onOperatorApplicationStateChange` and `onOperatorApplicationStateChangeCompleted` functions; and After the timer expires, the operator application shall be moved to background state again, with the Application being notified via an event dispatched to the `onOperatorApplicationStateChange` and `onOperatorApplicationStateChangeCompleted` functions. If an operator application calls the `opAppRequestForeground` method while the timer is running and the request is granted, the timer shall

	be disabled. If an operator application calls the `opAppRequestBackground` method before the timer expires the timer shall be disabled. If the request is not in accordance with the requirements listed in clause 6.3.3.4 of the present document then the request shall not be granted and the method shall return `false`. Note that the purpose of the timer is to prevent malfunctioning or misbehaving operator applications from remaining in transient state for prolonged periods of time. Operator applications should not rely on the timer running out as the primary means for moving to background state, but should call the `OpAppRequestBackground` method instead.

`Boolean opAppRequestForeground()`	
Description	Requests the terminal to move the calling application to foreground state. The terminal shall only grant the request if it was received in accordance with the specifications of clause 6.3.3.2 of the present document. When the request is granted, the Application shall be notified via an event dispatched to the `onOperatorApplicationStateChange` and `onOperatorApplicationStateChangeCompleted` functions and the method shall return `true`. If the request is not in accordance with the requirements listed in clause 6.3.3.2 of the present document then the request shall not be granted and the method shall return `false`.

`void opAppRequestBackground()`	
Description	Requests the terminal to move the calling application to background state. When the request is granted, the Application shall be notified via an event dispatched to the `onOperatorApplicationStateChange` and `onOperatorApplicationStateChangeCompleted` functions.

`void opAppRequestUpdate(Boolean immediate, [String params])`		
Description	When called by an operator application, the terminal attempts to update the operator application. The process of updating the app is asynchronous.	
Arguments	immediate	If true, the update shall be initiated immediately if the operator application is in the foreground state. If the operator application is in any other state, the terminal should instead perform an update as if immediate was set to false. If the terminal and the operator application are able to perform the entire update process (including operator application reboot) without interrupting linear video and audio then the bilateral agreement may define exceptions to the preceding statements, allowing for immediate updates in other states. If false, the update should occur at a time convenient for the user (e.g. when the terminal is in standby for a significant period). If called more than once then the most recent request shall take precedence and all previous ones shall be discarded. Where the call results in an update being initiated immediately, then when this method returns, subsequent calls to opAppUpdateStatus() shall immediately reflect the status of this update.
	params	An optional argument that, if used, shall be added to the end of the request for the XML AIT as defined in clause 6.1.5.1. The parameters need to be URL friendly (see RFC 3986 [i.7] and any encoding is the responsibility of the operator application.

`Integer opAppUpdateStatus()`	
Description	Returns the current status of any ongoing activity to update this operator application. The value returned by this function shall be:

Value	Description
-3	Discovery of updates is in progress.
-2	No update is in progress and either • there was no previous update, or • the previous update rediscovered the installed version of the operator application software, and no valid updates were signalled in the (XML) AIT(s), or • the previous update was before this operator application (re)started, and this operator application was not started with the status launch parameter value "updateFailed" (see section 7.1.2).
-1	New software is available to download for the operator application but download has not yet started.
0 - 99	New software for the operator application is being downloaded to the terminal and the value gives an approximation of the amount already downloaded.
100	Indicates that new software for the operator application has been successfully downloaded to the terminal and is about to be installed.
1 001 - 1 999	Indicates that an error occurred during the download of new software for the operator application to the terminal. This range of values can be used to provide an implementation specific error code defined in the bilateral agreement.
2 000 - 2 999	Indicates that an error occurred during the installation of new software for the operator application in the terminal after a successful download. This range of values can be used to provide an implementation specific error code defined in the bilateral agreement.
3000 - 3999	Indicates that an error occurred during the discovery of new software for the operator application in the terminal. This range of values can be used to provide an implementation specific error code defined in the bilateral agreement.

`Application createApplication( String uri, Boolean createChild, Boolean runAsOpApp )`		
Description	Create a new application. This call is asynchronous and will return before the new application is fully loaded. Calling this method does not automatically show the newly-created application. If the application cannot be created, this method shall return `null`. Behaviour if the application can be created but subsequently fails to load shall be as defined by clause 6.2.2.5.6 of ETSI TS 102 796 []. If a regular HbbTV® application calls this method with *runAsOpApp* being `true` or *createChild* being `true` then the method shall return null. If a regular HbbTV® application calls this method with *runAsOpApp* being `false` and *createChild* being `false` then this shall be identical to calling the method without either argument. Launching a new application to run as an operator application (*runAsOpApp* being `true`) shall replace the calling operator application with a new operator application in the same way as a regular HbbTV® application calling `createApplication( String uri, false )` is replaced by the new regular HbbTV® application. Operator applications distributed via broadband shall be referenced using HTTPS URLs that refer to an XML AIT as defined for regular HbbTV® broadcast-independent applications, with the additional requirement that the <mhp:URLBase> element in the XML AIT shall reference an HTTPS URL. Operator applications installed in the terminal shall be referenced as defined in clause 9.4.2 of the present document. Launching a new application to run as a broadcast-independent regular HbbTV® application (*runAsOpApp* being `false`) shall replace any running regular HbbTV® application with the new broadcast-independent regular HbbTV® application as defined in clause 6.2.2.6.1 of ETSI TS 102 796 []. The calling operator application shall continue running without interruption but shall be forced into the background state once the newly launched application has successfully been loaded. If an operator application launches a broadcast-independent regular HbbTV® application with *createChild* being true then it shall be notified with an `ApplicationLoaded` event when the launched application loads successfully and with an `ApplicationUnloaded` event when it exits.	
Arguments	*uri*	The URI of the application to launch.
	createChild	When *runAsOpApp* is false, defines whether the launching application is to be notified when the launched application exits (`true`) or not (`false`). This shall be ignored when *runAsOpApp* is true.
	runAsOpApp	This optional argument defines whether the application shall be launched as an operator application (`true`) or a broadcast-independent regular HbbTV® application (`false`). If the argument is not included then the value shall be considered to be false.

`Boolean opAppUninstall()`	
Description	Requests the terminal to uninstall the calling operator application. If the calling application cannot be uninstalled then `false` shall be returned. Otherwise the method shall return `true`. Some examples of reasons why the calling application cannot be uninstalled include the following: • The calling application is not an operator application. • The calling application is an operator application that is not installed (e.g. one running over broadband). • The calling application is an operator-specific operator application and the terminal cannot be used without it installed. If the method returns `true` then the calling application should immediately call `Application.destroyApplication()` and not attempt to load any more files from the installation (HTML, JavaScript, style sheets, images, …) as these may no longer be available.

`Storage getPrivateLocalStorage()`	
Description	Requests the terminal to return a Storage object referring to private persistent storage for the origin of the operator application. The Storage object shall support the Storage API interface as defined in [20] ,Section 4.1.

`String getOpApp2AppBaseURL()`	
Description	When called by an operator application, this method shall return the base URL of the local "operator application endpoint" that is used for communicating with regular HbbTV® applications, as defined in clause 15. The URL retrieved by this method shall end with a slash ('/') character. If called by a regular HbbTV® application the return value shall be null.

`String getApp2OpAppBaseURL()`	
Description	When called by a regular HbbTV® application, this method shall return the base URL of the local "regular application endpoint" that is used for communicating with operator applications as defined in clause 15. The URL retrieved by this method shall end with a slash ('`/`') character. If called by an operator application the return value shall be null.

A.2.2.3 Events

The Application class shall be extended with the following event.

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
`onOperatorApplicationStateChange`	`OperatorApplicationStateChange`	Bubbles: No Cancellable: No Context Info: `oldState, newState`
`onOperatorApplicationStateChangeCompleted`	`OperatorApplicationStateChangeCompleted`	Bubbles: No Cancellable: No Context Info: `oldState, newState`
`onOperatorApplicationContextChange`	`OperatorApplicationContextChange`	Bubbles: No Cancellable: No Context Info: `startupLocation, launchLocation`
`onOppAppUpdate`	`OpAppUpdate`	Bubbles: No Cancellable: No Context Info: `updateEvent`

The DOMs event 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 class. The third parameter of addEventListener, i.e. "useCapture", will be ignored.

A.2.3 ChannelConfig class

In the definition of the startScan method, the following requirement:

- 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.

Shall be replaced with the following:

- Start a scan for new channels on all available sources. When each source finishes scanning, a `ChannelListUpdate` event shall be raised and any channel lists for that source shall have been updated.

The `ChannelConfig` class shall be extended with the following methods;

`BroadcastSupervisor getBroadcastSupervisor ()`	
Description	When called by an operator application, this method shall return the `BroadcastSupervisor` instance for that operator application. Calls to this method by the same running application shall always return the same instance. When read by a regular HbbTV® application, this property shall return `undefined`.

`void setChannelList ( ChannelList list)`		
Description	Sets the channel list to be used for UI with the user, for regular HbbTV® applications, for the calling operator application and for any operator application started by the calling operator application. This method provides an alternative to an RF-based channel scan for configuring the channel list. Channels in this channel list may be ones created by the terminal or locally defined channels created from information obtained via IP, from a CICAM or from a broadcast carousel in a known channel. Channel lists set by this method shall only be retained either until replaced by a later call to this method by the same application or until the calling operator application and any operator application started by the calling application exits. If an operator application exits without starting any other operator applications and is later restarted, the channel list returned by the channelList property on the video/broadcast object and the ChannelConfig class shall be the original terminal channel list (potentially updated) and not the one set through this method when the operator application was previously running. The original terminal channel list is no longer accessible to an operator application once it has called this method. Operator applications that wish to include channels from the original terminal channel list in the channel list they pass to this method need to retain that channel list within the application.	
Arguments	`list`	The new channel list for the terminal.

`ChannelList createChannelList ( Channel[]  channels)`		
Description	Creates a ChannelList from an array of Channel objects. Any entries in the array that are not Channel objects shall be discarded.	
Arguments	`channels`	An array of Channel objects.

The `createChannelList(String)` method defined in OIPF DAE clause 7.13.9.2 shall be generalised to any XML description of a channel list as follows:

`ChannelList createChannelList ( String bdr)`		
Description	Creates a ChannelList object from an XML metadata format such as an SD&S Broadcast Discovery Record or DVB-I Service Discovery document. Channels in the returned channel list will not be included in the channel list that can be retrieved via calls to getChannelConfig().	
Arguments	`bdr`	An XML-encoded string containing channel metadata such as an SD&S Broadcast Discovery Record as specified in [OIPF_META2] or DVB-I Service Discovery document. If the string is not valid or the format is not supported by the terminal, this method shall return null.

NOTE: See clause 7.6 for the definition of the metadata formats required by the present document.

A.2.4 Operator applications and the video/broadcast object

A.2.4.1 Modifications to the state machine

The state machine for the `video/broadcast` object as defined clause 7.13 of the OIPF DAE specification [2] is modified as follows for operator applications.

- When an operator application transitions to either the background, transient or overlaid transient state, any video/broadcast object not in the unrealized state shall transition to that state. If the video/broadcast object had been in the presenting state prior to the state transition, any content being presented by that object shall continue to be presented under the control of the terminal regardless of the source of the content, e.g. RF tuner or broadband.

NOTE: This requirement applies regardless of whether the content is in the terminal channel list or is a locally defined channel.

- While an operator application is in either the background, transient or overlaid transient state, the video/broadcast object shall work as specified except that transitions from the unrealized state shall be blocked.

- When an operator application presented channel is selected, clause 9.9.5 shall apply.

A.2.4.2 Modification to onChannelChangeSucceeded

The `onChannelChangeSucceeded` function and the corresponding `ChannelChangeSucceeded` event are extended as shown underlined.

`function onChannelChangeSucceeded( Channel channel, Channel viewerChannel, Number quiet )`
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 terminal (see clause 7.13.1.1 of the OIPF DAE specification [2]). For an operator application, the specified function shall be always called with 3 arguments 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 clause 7.13.7 of the OIPF DAE specification [2]). • `Channel viewerChannel` - the channel to be used for all channel-related interaction by the viewer, i.e. the most recent channel to which there was a successful channel change excluding ones with `quiet` = 2. • `Number quiet` - Flag indicating whether the channel change operation is to be carried out quietly, as described in clause A.2.4.3.2 of ETSI TS 102 796 [].The operator application shall act on the value as specified including the following: – not drawing the channel banner when it is not supposed to be drawn – only using the new channel as the basis for future relative channel changes (i.e. using P+ / P-) when it is supposed to be used (i.e. when `quiet` = 0 and `quiet` = 1, but not when `quiet` = 2) – if any front panel channel display would be controlled by the operator application using the APIs for access to proprietary functions (see clause 8.3.1 of the present document) then ensuring the channel displayed is not updated when `quiet` = 2.

A.2.5 The BroadcastSupervisor class

The BroadcastSupervisor class enables an operator application to monitor and partly control broadcast video presentation regardless of the operator application state as long as broadcast TV is active (i.e. excluding the opapp-silent context). An operator application shall be able to obtain an instance of the BroadcastSupervisor class from the `ChannelConfig.getBroadcastSupervisor` method.

The properties, events and methods on this class are largely defined in terms of those on the video/broadcast object in the connecting or presenting state (as appropriate). They shall reflect the broadcast content being presented, regardless whether this content is being presented by a video/broadcast object in a regular HbbTV® application, by a video/broadcast object in the operator application or by the terminal itself.

If the terminal is presenting the broadcast content, it is assumed that the terminal software has equivalent properties to those on a video/broadcast object (e.g. playState).

If the regular HbbTV® application or the operator application has a video/broadcast object not in the unrealized state, all of the events listed in table A.5 shall be dispatched to the video/broadcast object (if appropriate) as well as the BroadcastSupervisor object.

Table A.5: Properties and Events of the BroadcastSupervisor class

Property name	Behaviour for `BroadcastSupervisor` class
onChannelChangeError property and ChannelChangeError event	Shall be called/generated as specified for a video/broadcast object for channel changes regardless of how those channel changes were initiated except as follows. Error code 3, 'parental lock on channel', is split (only for the `BroadcasSsupervisor` object). Error code '3' shall be used only when a channel is blocked due to signalling associated with the channel. A new error code '13' shall be used when a channel is blocked due to manual action by the user.
playState	Shall return one of the following: • the same value as the property of the same name on the video/broadcast object in the regular HbbTV® application, if one exists that is not in the unrealized state; otherwise; • the same value as the property of the same name on the video/broadcast object in the operator application, if one exists that is not in the unrealized state; otherwise;

Property name	Behaviour for `BroadcastSupervisor` class
	• unrealized if the operator application is in the opapp-silent context, otherwise; • the equivalent state of the terminal software if the terminal is presenting broadcast content to the user.
playStateError	Shall return the error code as specified for a video/broadcast object, whenever the playState property on the BroadcastSupervisor object changes value and the corresponding PlayStateChange event has an error parameter that is not undefined. If no error occurred, this property shall return undefined. The possible values shall be the same as listed in [2] section 7.13.1.2 `onChannelChangeError()` of the errorState property. If an error condition that occurred before the operator application started persists once the operator application is running then this property shall be set appropriately for the persisting error condition. An example of a persisting error condition would be if the bilateral agreement defined that the terminal would always block video and audio on power-on or on switching to linear TV mode so that the operator application would have to unblock them after it started.
onPlayStateChange property and PlayStateChange event	Shall be called/generated as specified for a video/broadcast object whenever the playState property on the BroadcastSupervisor object changes value except as follows. If a channel is selected that has been manually blocked by the user, the error argument shall be set to 13 and not to 3 as is the case for a video/broadcast object.
onChannelChangeSucceeded property and ChannelChangeSucceeded event	Shall be called/generated as specified for a video/broadcast object for channel changes regardless of how those channel changes were initiated.
onPlaySpeedChanged property and PlaySpeedChanged event	Shall be called/generated as specified for a video/broadcast object whenever the playSpeed property on the BroadcastSupervisor object changes value.
onPlayPositionChanged and PlayPositionChanged event	Shall be called/generated as specified for a video/broadcast object whenever the playPosition property on the BroadcastSupervisor object changes value and that change occurs due to the use of trick play functions.
playbackOffset	Shall return the same value as the property of the same name on either the video/broadcast object in the running regular HbbTV® application or the video/broadcast object in the operator application or the equivalent property in the terminal software, whichever is currently presenting the broadcast content.
maxOffset	Shall return the same value as the property of the same name on either the video/broadcast object in the running regular HbbTV® application or the video/broadcast object in the operator application or the equivalent property in the terminal software, whichever is currently presenting the broadcast content.
recordingState	Shall return the same value as the property of the same name on either the video/broadcast object in the running regular HbbTV® application or the video/broadcast object in the operator application or the equivalent property in the terminal software, whichever is currently presenting the broadcast content.
onRecordingEvent property and RecordingEvent	Shall be called/generated as specified for a video/broadcast object whenever the recordingState property on the BroadcastSupervisor object changes value.
playPosition	Shall return the same value as the property of the same name on either the video/broadcast object in the running regular HbbTV® application or the video/broadcast object in the operator application or the equivalent property in the terminal software, whichever is currently presenting the broadcast content.
playSpeed	Shall return the same value as the property of the same name on either the video/broadcast object in the running regular HbbTV® application or the video/broadcast object in the operator application or the equivalent property in the terminal software, whichever is currently presenting the broadcast content.
playSpeeds[]	Shall return the same value as the property of the same name on either the video/broadcast object in the running regular HbbTV® application or the video/broadcast object in the operator application or the equivalent property in the terminal software, whichever is currently presenting the broadcast content.
onPlaySpeedsArrayChanged and PlaySpeedsArrayChanged event	Shall be called/generated as specified for a video/broadcast object whenever the playSpeeds[] property on the BroadcastSupervisor object changes.
timeShiftMode	Shall return the same value as the property of the same name on either the video/broadcast object in the running regular HbbTV® application or the video/broadcast object in the operator application or the equivalent property in the terminal software, whichever is currently presenting the broadcast content.

Property name	Behaviour for `BroadcastSupervisor` class
currentTimeShiftMode	Shall return the same value as the property of the same name on either the video/broadcast object in the running regular HbbTV® application or the video/broadcast object in the operator application or the equivalent property in the terminal software, whichever is currently presenting the broadcast content.
programmes	Shall return one of the following: • the same value as the property of the same name on the video/broadcast object in the regular HbbTV® application, if one exists that is not in the unrealized state; otherwise; • the same value as the property of the same name on the video/broadcast object in the operator application, if one exists that is not in the unrealized state; otherwise; • the collection of programmes, as defined in OIPF [2] for the property of the same name, available on the currently tuned channel, if one is being presented by the terminal; otherwise; • a collection of length 0.
onProgrammesChanged property and ProgrammesChanged event	Shall be called/generated as specified for a video/broadcast object whenever the programmes property on the BroadcastSupervisor object changes.
onParentalRatingChange property and ParentalRatingChange event	Shall be called/generated as specified for a video/broadcast object during playback of a channel, however it is being presented.
onParentalRatingError property and ParentalRatingError event	Shall be called/generated as specified for a video/broadcast object during playback of a channel, however it is being presented.
onDRMRightsError property and DRMRightsError event	Shall be called/generated as specified for a video/broadcast object during playback of a channel, however it is being presented.
currentChannel	Shall return the value of the property of the same name as specified for a video/broadcast object, however the current channel is being presented.
onSelectedComponentChanged property or SelectedComponentChange event	Shall be called/generated as specified for a video/broadcast object during playback of a channel, however it is being presented.
onApplicationLoaded property and ApplicationLoaded event	Shall be called/generated when a regular HbbTV application associated with a channel is started successfully. See clause 6.5.2 for constraints on the application object method argument.
onApplicationLoadError property and ApplicationLoadError event	Shall be called/generated when a regular HbbTV application associated with a channel fails to start successfully. See clause 6.5.2 for constraints on the application object method argument.

Table A.6: Methods of the BroadcastSupervisor class

Method name	Behaviour
getChannelConfig	Shall work as specified for a video/broadcast object.
createChannelObject(Integer idType, String dsd, Integer sid)	Shall work as specified for a video/broadcast object.
createChannelObject(Integer idType, Integer onid, Integer tsid, Integer sid, Integer sourceID, String ipBroadcastID)	Shall work as specified for a video/broadcast object.
setChannel(Channel channel, Boolean trickplay, String contentAccessDescriptorURL)	Shall work as specified for a video/broadcast object except that references to the video/broadcast object shall be interpreted as meaning the presentation of broadcast video by whatever means are currently being used (see above text in this clause).
setChannel(Channel channel, Boolean trickplay, String contentAccessDescriptorURL, Number quiet, Boolean blockAV)	Shall work as specified for a video/broadcast object except as follows; 1) references to the video/broadcast object shall be interpreted as meaning the presentation of broadcast video by whatever means are currently being used (see above text in this clause). 2) the optional `blockAV` argument, if set to true, shall result in the media components of the newly selected channel being blocked immediately the channel is selected without ever being shown. If set to false or absent, the media components of the selected channel shall not be blocked. Setting this to true is different from calling `blockAV`() in the handler for a `ChannelChangeSucceeded` event as this shall not result in any media

Method name	Behaviour
	components being visible or audible, even transiently. Once the terminal has blocked all the components, the terminal shall; • Raise a PlayStateChange event with the property state set to 1 (connecting) and error set to 3 (parental lock on channel). • Set the property playStateError to 3 (parental lock on channel) on the BroadcastSupervisor object
setChannel(null)	Shall work as specified for a video/broadcast object except that the calling application shall not transition to a broadcast-independent application, specifically the requirement in clause A.2.4.1 of TS 102 796 [1] shall not apply. Any running broadcast-related application shall be terminated instead of transitioning to be broadcast-independent. The calling Operator Application shall continue to run without becoming broadcast-independent. NOTE: Privileged and operator-specific operator applications are neither broadcast-related nor broadcast-independent and cannot transition to become either of them.
prevChannel(Boolean blockAV)	Shall work as specified for a video/broadcast object except as follows; 1) that calls to this method are always valid and that references to the video/broadcast object shall be interpreted as meaning the presentation of broadcast video by whatever means are currently being used (see above text in this clause). 2) the optional `blockAV` argument, if set to true, shall result in the media components of the newly selected channel being blocked immediately the channel is selected without ever being shown. If set to false or absent, the media components of the selected channel shall not be blocked. Setting this to true is different from calling `blockAV`() in the handler for a `ChannelChangeSucceeded` event as this shall not result in any media components being visible or audible, even transiently. Once the terminal has blocked all the components, the terminal shall; • Raise a PlayStateChange event with the property state set to 1 (connecting) and error set to 3 (parental lock on channel). • Set the property playStateError to 3 (parental lock on channel) on the BroadcastSupervisor object
nextChannel(Boolean blockAV)	Shall work as specified for a video/broadcast object except as follows; 1) that calls to this method are always valid and that references to the video/broadcast object shall be interpreted as meaning the presentation of broadcast video by whatever means are currently being used (see above text in this clause). 2) the optional `blockAV` argument, if set to true, shall result in the media components of the newly selected channel being blocked immediately the channel is selected without ever being shown. If set to false or absent, the media components of the selected channel shall not be blocked. Setting this to true is different from calling `blockAV`() in the handler for a `ChannelChangeSucceeded` event as this shall not result in any media components being visible or audible, even transiently. Once the terminal has blocked all the components, the terminal shall; • Raise a PlayStateChange event with the property state set to 1 (connecting) and error set to 3 (parental lock on channel). • Set the property playStateError to 3 (parental lock on channel) on the BroadcastSupervisor object
recordNow()	Shall work as specified for a video/broadcast object as if the current object were the video/broadcast object currently presenting the broadcast channel, if one is being presented by the regular HbbTV® application or the terminal.
stopRecording()	Shall work as specified for a video/broadcast object when recordNow() was called on the current object
pause()	Shall work as specified for a video/broadcast object as if the current object were the video/broadcast object currently presenting the broadcast channel, if one is being presented by the regular HbbTV® application or the terminal.
resume()	Shall work as specified for a video/broadcast object when pause() was called on the current object.
setSpeed(Number speed)	Shall work as specified for a video/broadcast object as if the current object were the video/broadcast object currently presenting the broadcast channel, if one is being presented by the regular HbbTV® application or the terminal.

Method name	Behaviour
seek(Integer offset, Integer reference)	Shall work as specified for a video/broadcast object as if the current object were the video/broadcast object currently presenting the broadcast channel, if one is being presented by the regular HbbTV® application or the terminal.
stopTimeshift()	Shall work as specified for a video/broadcast object as if the current object were the video/broadcast object currently presenting the broadcast channel, if one is being presented by the regular HbbTV® application or the terminal.
getComponents(Integer componentType)	Shall work as specified for a video/broadcast object except that the set of components shall be known if broadcast content is being presented.
getCurrentActiveComponents(Integer componentType)	Shall work as specified for a video/broadcast object except that the set of components shall be known if broadcast content is being presented.
selectComponent(AVComponent component)	Shall work as specified for a video/broadcast object.
unselectComponent(AVComponent component)	Shall work as specified for a video/broadcast object.
selectComponent(Integer componentType)	Shall work as specified for a video/broadcast object.
unselectComponent(Integer componentType)	Shall work as specified for a video/broadcast object.
blockAV()	Only defined for the broadcast supervisor and not for the video/broadcast object.
unblockAV()	Only defined for the broadcast supervisor and not for the video/broadcast object.
addEventListener(type, listener)	Shall be called as defined for a video/broadcast DOM object. The method shall attach the specified listener to the BroadcastSupervisor instance that will be called whenever the specified event type defined in Table A.5 is generated. Returns undefined.
removeEventListener(type, listener)	Shall be called as defined for a video/broadcast DOM object. The method shall remove an event listener previously registered with the BroadcastSupervisor.addEventListener() method. The event listener to be removed is identified using a combination of the event type and the event listener function itself. Calling BroadcastSupervisor.removeEventListener() with arguments that do not identify any currently registered event listener shall have no effect. Returns undefined.

The BroadcastSupervisor object shall be extended with the following methods.

`void blockAV()`	
Description	This method shall block all media components including subtitles on the current channel. If the channel was previously unblocked, then once the terminal has blocked all the components, the terminal shall; • Raise a `PlayStateChange` event with the property `state` set to 1 (connecting) and `error` set to 3 (parental lock on channel). • Set the property `playStateError` to 3 (parental lock on channel) on the BroadcastSupervisor object. Operator applications shall be able to call this method when the broadcast supervisor is in the presenting or connecting state. Calling this method when the broadcast supervisor is in other states shall fail silently.

`void unblockAV()`

Description	The terminal shall unblock media components on the current channel including subtitles.The media components presented shall be as defined in clause 10.2.7 ot TS 102 796 [1]. Once the terminal has unblocked the current channel, the terminal shall; • raise a `PlayStateChange` event with the property `state` set to 2 (presenting) • set the `playStateError` property to `undefined` on the BroadcastSupervisor object Operator applications shall be able to call this method whilst the current channel is in a connecting state and playStateError property is set to 3. Calling this method when the preceding conditions do not apply (e.g. on an unblocked channel) shall fail silently.

A.2.6 oipfDrmAgent

A.2.6.1 Shared use of DRM and Conditional Access messaging in the terminal

The application/oipfDrmAgent embedded object allows an application to send messages to and receive messages from a security module (DRM or Conditional Access) in the terminal. The OIPF DAE Specification [2] defines how the terminal determines where to route messages originating from an application (sent using the sendDRMMessage method) and, when only one application is running or there is only one application/oipfDrmAgent embedded object, it is obvious how the terminal routes messages being sent back by the security module.

When an operator application is running at the same time as a regular HbbTV® application and both applications have an application/oipfDrmAgent embedded object, the following rules shall apply:

- Messages sent using the sendDRMMessage method shall be routed inside the terminal as defined by the OIPF DAE Specification [2].
- Messages received from the security module that are in reply to a message sent using the sendDRMMessage message shall be passed to the application that sent the original message.
- Messages received from the security module that are not in reply to a message sent using the sendDRMMessage message shall be passed to both the regular HbbTV® application and the operator application.

NOTE: DRM systems should avoid sending regular HbbTV® applications DRMSystem messages that they have not been tested with. If a DRM system uses messages that are only for an operator application, then a different DRM system ID should be used to avoid this.

A.2.7 oipfRecordingScheduler

If the terminal supports the PVR feature, the application/oipfRecordingScheduler embedded object allows applications to schedule recordings, retrieve the list of recordings (scheduled, in progress, and completed), and remove scheduled recordings.

- For both privileged operator applications and operator-specific operator applications, the following shall apply:
    - The `getScheduledRecordings` method shall return all recordings that are scheduled but which have not yet started regardless of the origin of the application that scheduled them.

NOTE: The above requirement is an omission in the OIPF DAE specification where this method has the same behaviour regardless of the value of the "manageRecordings" capability.

- The `getInProgressRecordings` method shall return all recordings that are currently in progress regardless of the origin of the application that scheduled them.

- The `recordAt` method shall set a manual recording and return a `ScheduledRecording` object containing the `scheduleId, startTime, duration, channelID` and `repeatDays` as specified in the call to the method.
- The `update` method shall modify a scheduled recording where the argument `id` shall be set to the `scheduleId` property of a `ScheduledRecording` object. When a scheduled recording has not yet started, it shall be possible for applications to update the `startTime`, `duration` and the `repeat` properties. Once recording has started, it shall only be possible for applications to update the `duration` field.

- For operator-specific operator applications (and not privileged operator applications), the following shall apply:
  - The `remove` method shall be able to remove all scheduled, in-progress or completed recordings regardless of the origin of the application that scheduled them.
- For privileged operator applications the following shall apply:
  - The `remove` method shall be able to remove scheduled, in-progress or completed recordings when the recording was scheduled by applications from the same origin as the caller.

A.2.8 Extensions to the application/oipfParentalControlManager object

A.2.8.1 Properties

The application/oipfParentalControlManager object shall be extended with the following property.

`readonly Integer parentalPINLength`
The number of digits that make up the parental control PIN in the terminal. Shall return 0 if parental access control is disabled (e.g. explicitly by the user). Shall return -1 if parental access control is enabled but does not use a PIN code (e.g. uses a gesture or voice input).

A.2.8.2 Methods

The `application/oipfParentalControlManager` class shall be extended with the following method.

`Promise requestParentalControlApproval(Object context)`	
Description	Run the terminal's process for approving access for content that is blocked by parental access control. The method is asynchronous. If the terminal uses a PIN code the terminal will prompt for that PIN code and validate it. If the terminal uses some other mechanism (e.g. a gesture or voice input) then the terminal will prompt for that and validate it. Returns a `Promise` which indicates if the access was approved or not. If access to the blocked content was approved then the promise shall be resolved with value “approved”. If access to the blocked content was not approved (either explicitly or implicity through a timeout or repeated failure to input a correct PIN code) then the promise shall be resolved with value “notApproved”.
Arguments	Either null or an object containing a set of key-value pairs where each key shall be a BCP-47 [27] language code and each value is a human-readable name of the content to be unblocked in the language of the key that will be shown to the user if the terminal needs one. If an object is passed then it should include at least one key-value pair where the key is the same BCP47 primary language subtag as `Configuration.preferredUILanguage47`.

A.2.8.3 Clarifications

The following clarifications shall apply to the the `unlockWithParentalControlPIN` method;

- If the target argument of the `unlockWithParentalControlPIN` method is an HTML5 media element using the native DASH player, the content being presented through this object shall be unlocked until a new item of content is played using this object.

- If the target argument of the `unlockWithParentalControlPIN` method is an HTML5 media element using Media Source Extensions, a `DOMException` with error name "InvalidStateError" shall be returned.
- If the target argument of the `unlockWithParentalControlPIN` method is a `BroadcastSupervisor` object, the content being monitored through this object shall be unlocked until a new channel is selected.

A.2.9 Extensions to the Channel class

A.2.9.1 Properties

The Channel class shall be extended with the following property.

`readonly Integer tunerID`
The unique identifier of the tuner through which this channel is being received. This value shall be valid and correct when the video/broadcast object being used to present this channel is in the Connecting, Presenting or Stopped state and the channel is not an IP channel.

Annex B (normative): HbbTV® use of the DVB URI_linkage_descriptor

B.1 Introduction

ETSI EN 300 468 [14] defines a URI_linkage_descriptor which allows a broadcaster to insert a URI in the broadcast stream. HbbTV® makes use of this descriptor in the present document to carry location information used during the discovery of operator applications. To this end, HbbTV® has registered a value of the uri_linkage_type which allows terminals to determine which URI_linkage_descriptors conform to the profile defined in this annex. The value assigned by DVB is 0x60.

Future documents written by HbbTV® may use this descriptor for other use cases.

B.2 URI_linkage_descriptor profile

When the uri_linkage_type is set to the value registered by HbbTV® (i.e. 0x60), the value of the private_data_bytes shall be as defined in table B.1.

Table B.1: HbbTV® use of private_data_bytes

Syntax	# of bits	Mnemonic
`hbbtv_linkage_type`	4	uimbsf
`reserved`	3	bslbf
`polling_flag`	1	bslbf
`if (polling_flag == 1) {`		
`  min_polling_interval`	16	uimbsf
`}`		
`for (i=0;i<N;i++) {`		
`  hbbtv_private_data_byte`	8	uimbsf
`}`		

Semantics:

hbbtv_linkage_type: This field is a 4-bit field specifying the type of URI linkage. HbbTV terminals shall ignore values which they have not been designed to handle. It shall be encoded according to table B.2.

Table B.2: hbbtv_linkage_type field

hbbtv_linkage_type	Description
0x0	Operator application discovery, as defined in clause 6.1.2
0x1-0xB	Reserved for future use
0xC-0xF	User defined. Note: a private data specifier descriptor shall scope the URI_linkage_descriptor if the hbbtv_linkage_type is from the user defined range.

polling_flag: This 1-bit field if set to one indicates that the min_polling_interval field is present. This shall be set to zero when hbbtv_linkage_type is 0x0.

min_polling_interval: This 16-bit field is specified in ETSI EN 300 468 [14] in the definition of the URI_linkage_descriptor.

hbbtv_private_data_byte: This is an 8-bit field which is reserved for future use.

Annex C (informative): Sequence diagrams

C.1 Channel Change

The diagram in figure C.1 gives an example of how a privileged or operator-specific operator application handles a channel change initiated by the user using the P+ key on the remote control.

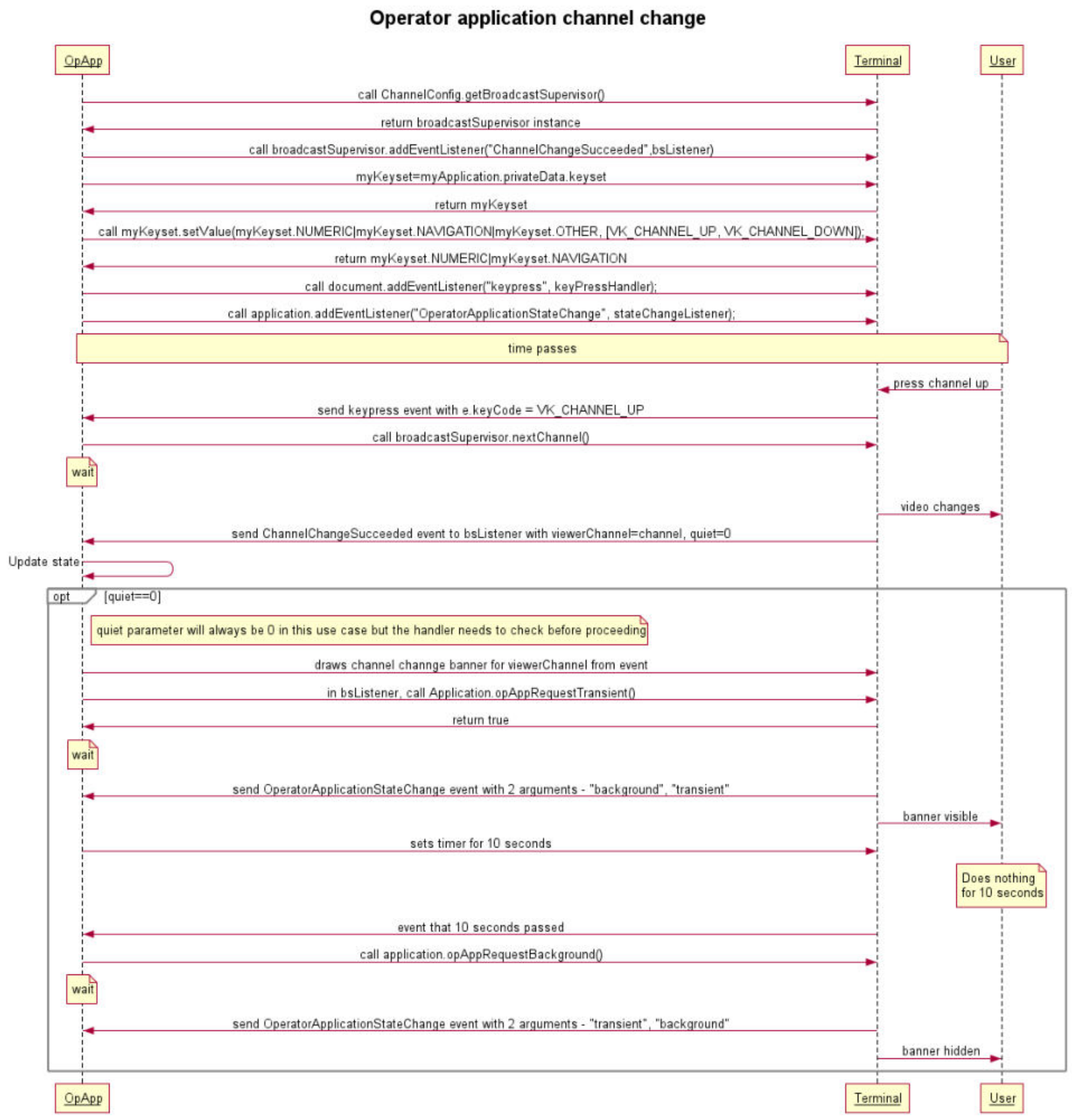
Operator application channel change
OpApp
Terminal
User
call ChannelConfig.getBroadcastSupervisor()
return broadcastSupervisor instance
call broadcastSupervisor.addEventListener("ChannelChangeSucceeded",bsListener)
myKeyset=myApplication.privateData.keyset
return myKeyset
call myKeyset.setValue(myKeyset.NUMERIC|myKeyset.NAVIGATION|myKeyset.OTHER, [VK_CHANNEL_UP, VK_CHANNEL_DOWN]);
return myKeyset.NUMERIC|myKeyset.NAVIGATION
call document.addEventListener("keypress", keyPressHandler);
call application.addEventListener("OperatorApplicationStateChange", stateChangeListener);
time passes
press channel up
send keypress event with e.keyCode = VK_CHANNEL_UP
call broadcastSupervisor.nextChannel()
wait
video changes
send ChannelChangeSucceeded event to bsListener with viewerChannel=channel, quiet=0
Update state
opt
[quiet==0]
quiet parameter will always be 0 in this use case but the handler needs to check before proceeding
draws channel channge banner for viewerChannel from event
in bsListener, call Application.opAppRequestTransient()
return true
wait
send OperatorApplicationStateChange event with 2 arguments - "background", "transient"
banner visible
sets timer for 10 seconds
Does nothing for 10 seconds
event that 10 seconds passed
call application.opAppRequestBackground()
wait
send OperatorApplicationStateChange event with 2 arguments - "transient", "background"
banner hidden
OpApp
Terminal
User

Figure C.1: Operator application channel change

C.2 Broadband-based operator application discovery

The following sequence diagrams shows an example of broadband-based operator application discovery.

The complete flow diagram is shown in figure C.1. In the example shown in figure C.2 it is assumed that the HbbTV® terminal is equipped with a hardwired FQDN named "operator.example.com". The domain name is used for preparing a DNS query by prefixing the domain name by "_hbbtv-ait._tcp".

In Step 1a) the terminal sends an SRV query to the DNS server of the service provider asking for the specification of the location of the server for the AIT information. This information is sent back in Step 1b) in a format that is described in IETF RFC 2782 [6]. Amongst others, this reply gives information about the symbolic name of the service, the transport protocol that is used and the assigned port number. In this example the AIT server can be reached under "ait.operator.example.com".

Using the AIT server name the HbbTV® terminal can construct now an HTTPS GET request by prefixing "https://" and appending "/opapp_ait.xml" to the SRV response. In Step 2a) the constructed GET command is sent to the identified AIT server and this server responds to the request in Step 2b) with the 200 OK message and the wanted XML file.

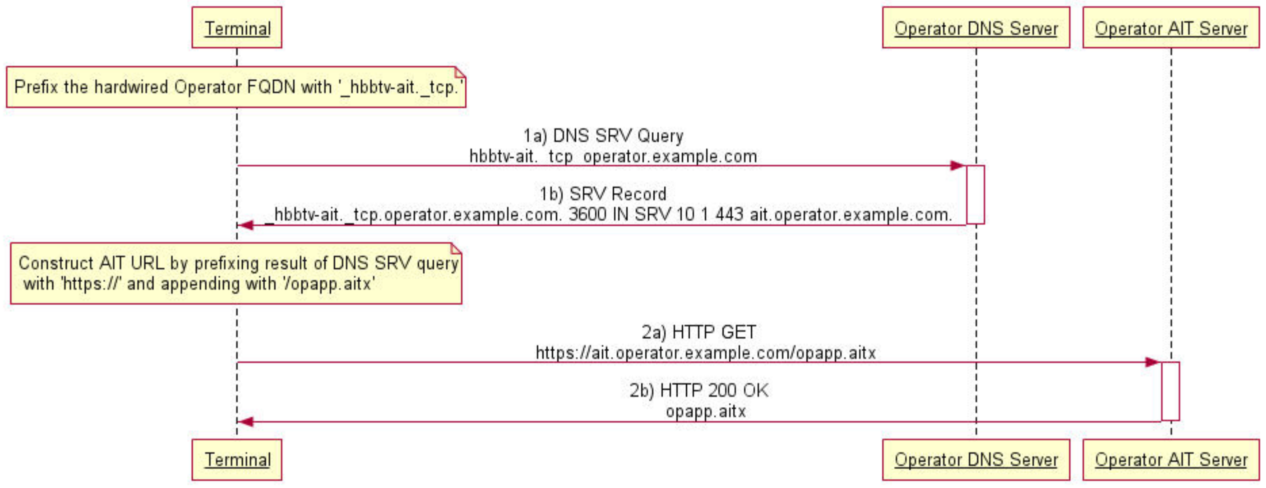
Terminal
Operator DNS Server
Operator AIT Server
Prefix the hardwired Operator FQDN with '_hbbtv-ait._tcp.'
1a) DNS SRV Query
hbbtv-ait._tcp_operator.example.com
1b) SRV Record
_hbbtv-ait._tcp.operator.example.com. 3600 IN SRV 10 1 443 ait.operator.example.com.
Construct AIT URL by prefixing result of DNS SRV query with 'https://' and appending with '/opapp.aitx'
2a) HTTP GET
https://ait.operator.example.com/opapp.aitx
2b) HTTP 200 OK
opapp.aitx
Terminal
Operator DNS Server
Operator AIT Server

Figure C.2: Operator application discovery from hardwired FQDN

C.3 Broadcast-based operator application discovery and installation

The example in figure C.3 shows an operator with an operator application having an organization identifier of 456 and an application identifier of 123. The encrypted package is called "enc-opapp.pkg" under a folder called "/release2" located in a DSM-CC carousel. This location is identified in the AIT carried in the service with the dvb triplet 50.2017.3000. The location of the AIT is identified in the NIT with a network identifier of 2 by a URI_linkage_descriptor.

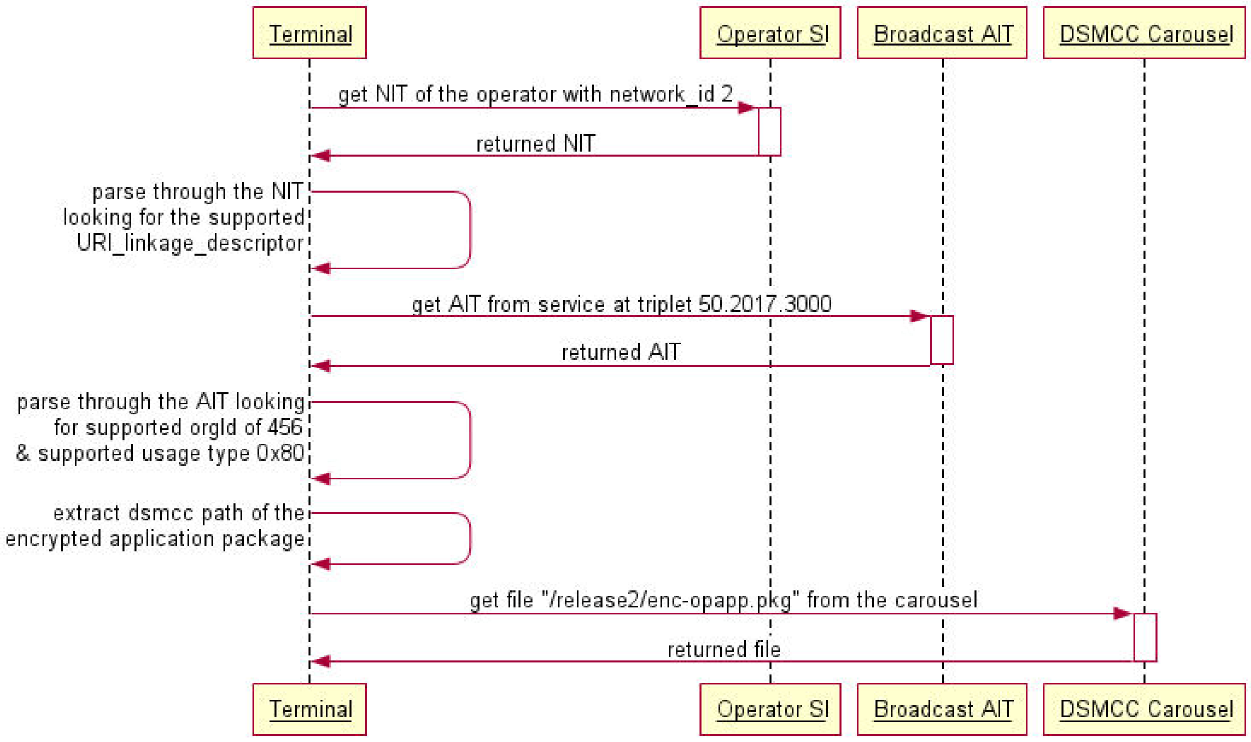
Terminal
Operator SI
Broadcast AIT
DSMCC Carousel
get NIT of the operator with network_id 2
returned NIT
parse through the NIT looking for the supported URI_linkage_descriptor
get AIT from service at triplet 50.2017.3000
returned AIT
parse through the AIT looking for supported orgId of 456 & supported usage type 0x80
extract dsmcc path of the encrypted application package
get file "/release2/enc-opapp.pkg" from the carousel
returned file
Terminal
Operator SI
Broadcast AIT
DSMCC Carousel

Figure C.3: Broadcast-based operator application discovery mechanism

C.4 Channel change to an operator application presented channel

The sequence diagram in Figure C.4 illustrates a channel change from a broadcast channel to an operator application presented channel.

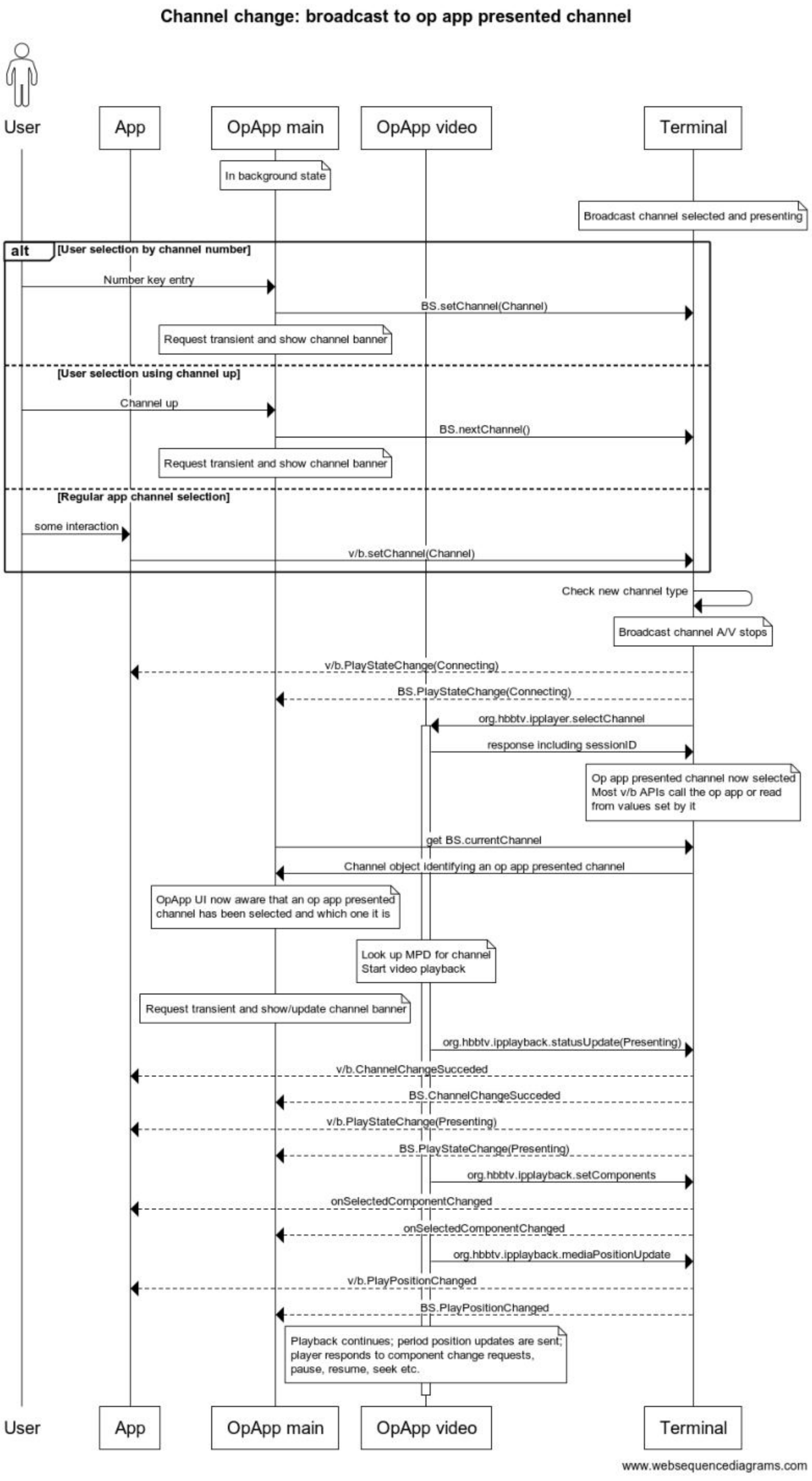
Channel change: broadcast to op app presented channel
User
App
OpApp main
OpApp video
Terminal
In background state
Broadcast channel selected and presenting
alt [User selection by channel number]
Number key entry
BS.setChannel(Channel)
Request transient and show channel banner
[User selection using channel up]
Channel up
BS.nextChannel()
Request transient and show channel banner
[Regular app channel selection]
some interaction
v/b.setChannel(Channel)
Check new channel type
Broadcast channel A/V stops
v/b.PlayStateChange(Connecting)
BS.PlayStateChange(Connecting)
org.hbbtv.ipplayer.selectChannel
response including sessionID
Op app presented channel now selected
Most v/b APIs call the op app or read
from values set by it
get BS.currentChannel
Channel object identifying an op app presented channel
OpApp UI now aware that an op app presented
channel has been selected and which one it is
Look up MPD for channel
Start video playback
Request transient and show/update channel banner
org.hbbtv.ipplayback.statusUpdate(Presenting)
v/b.ChannelChangeSucceded
BS.ChannelChangeSucceded
v/b.PlayStateChange(Presenting)
BS.PlayStateChange(Presenting)
org.hbbtv.ipplayback.setComponents
onSelectedComponentChanged
onSelectedComponentChanged
org.hbbtv.ipplayback.mediaPositionUpdate
v/b.PlayPositionChanged
BS.PlayPositionChanged
Playback continues; period position updates are sent;
player responds to component change requests,
pause, resume, seek etc.
User
App
OpApp main
OpApp video
Terminal
www.websequencediagrams.com

Figure C.4: Channel change from a broadcast channel to an operator application presented channel

Annex D (informative): Bilateral agreement

This annex summarizes recommended topics for the bilateral agreement between manufacturer and operator.

Terminals may make available to an operator more than what is required by the bilateral agreement with that operator. For example, they may make available to all operators the union of what is required by the bilateral agreement with each one of them. There is no requirement for a terminal to limit what is available to an operator to exactly what is defined in the bilateral agreement including that operator.

Table D.1: Recommended topics for bilateral agreement

Topic	Reference	Description
Business terms		
Quality assurance		The bilateral agreement may define business terms for testing both the operator application and the terminal.
Software updates		The bilateral agreement may include business terms for updating the operator application and for updating the system software.
Branding		The bilateral agreement may define business terms for how the terminal presents the operator application to the user.
Application provision		
organisation_id	6.1.5.1	The bilateral agreement needs to define the organisation_id of the operator.
Size of operator application	6.1.6	The bilateral agreement may define the maximum size of the uncompressed and extracted files of the operator application.
Type or types of operator application		The bilateral agreement needs to define whether it applies to privileged or operator-specific operator applications.
Discovery	6.1.2	The bilateral agreement needs to define which discovery methods are used for the operator application(s). Some discovery methods need extra information such as hardwired addresses or where to look for a NIT/BAT. If the operator application discovery process could result in more than one distinct multiple operator application from the same operator or more than one instance of the same operator application then the bilateral agreement needs to define how this is to be handled (e.g. user choice, one of them having priority, ...).
Non-connected operation		The bilateral agreement may define if the operator application needs an operational broadband connection.
Use and size of the private persistent storage	A.2.2.2	The bilateral agreement may define additional requirements on the operator on how the private storage is used and the information stored. The agreement may state an alternative maximum size. 1 Mbytes per origin is suggested as detailed in Table 11 under section 10.2.1 ETSI TS 102 796 [].
OpApp update in background state	6.2	If updating an operator application in the background state would not interrupt playback of linear video and audio then an exception concerning immediate updates in the background state may be made for the opAppRequestUpdate method.
Installation by the operator application after it has been installed by the terminal.	7.1.1	The bilateral agreement should define, when the operator application is installed by the terminal during the terminal's own install flow, if the terminal then launches the operator application to perform operator application specific installation in the "opapp-install" context. Alternatively the operator application may perform operator application specific installation on the first time it's run in normal use – in which case "opapp-install" context would not be used.
Application framework		
How the operator application is started	5.2.2 7.1.1	The bilateral agreement may define different entry points to the operator application.

Topic	Reference	Description
		• This may include the definition of different launch context signalling. • Depending on the power-on behaviour, the bilateral agreement may define that certain keys (e.g. EPG, channel list, a dedicated operator key) start the operator application when it is not already running.
Power-on behaviour	6.3.2.1	The bilateral agreement may define the power-on behaviour. This could include: • This may include the automatic restart of the operator application either after all reboots or • Restarting the operator appliocation only after reboots where the operator application was running previously, and/or • postponing starting the operator application when the terminal starts in a mode where linear TV is not in use.
Support for operator applications to run in active standby	6.3.4	The bilateral agreement shall define whether an operator application can continue to run during active standby if supported by the terminal.
Active standby period	6.3.4	The bilaterial agreement needs to define the period of time that an Operator Application may continue to run during active standby if supported by the terminal.
Failure of first-time installation	6.1.9.2	The bilateral agreement may define how failures during first-time installation are handled.
Icons	6.6.1 7.3.2 8.4.1	The bilateral agreement may define if the icons element in the XML AIT is used and if so; • the sizes of icons that are provided by the operator (see table 7: Definition of different icon flags of ETSI TS 102 809 [24]) • any optional purpose attributes.
Deciding which operator applications to install	6.1.6	The result of the previous processes is a number of (XML) AITs each corresponding to an operator application where a bilateral agreement is in place for the terminal. Which of these are installed depends on a number of factors. See clause 6.1.6.
Start pages of the operator application	5.3.2	The bilateral agreement may define different start pages of the operator application depending on the launch and/or startup contexts
Stopping the operator application	6.3.2.1 6.3.2.3	A number of ways are defined by which an operator application may be stopped. The bilateral agreement may define some conditions under which an operator application is required to be restarted but great care should be taken to avoid the user becoming trapped by an application error.
Visible RF-Based channel list	6.6.1	For IP discovered operator applications, which RF-based channel list or lists is visible to the operator application. For example, if the terminal has separate channel lists or operating modes, e.g. a terrestrial operating mode with its own channel list and a satellite operating mode with its own channel list.
Mandatory API	A.2	Some of those additional APIs may be made mandatory as part of a bilateral agreement. Clause A.2 defines modifications to the APIs from the OIPF DAE specification [2].
Web notifications	8.4.1	By default the user agent grants permission to display these for all origins used by operator applications for which it has a bilateral agreement - see clause 8.4.1.
Error codes	A.2.2.2.2.2	Indicates that an error occurred during the download of new software to the terminal. This range of values can be used to provide an implementation specific error code defined in the bilateral agreement.
Security		
Authentication and decryption of operator application	4.1.8 11.3.2	The bilateral agreement needs to address how: • the Operator Signing Root CA certificate (or an intermediate) is provided to the manufacturer. • the Terminal Packaging Certificate is provided to the operator.
Authentication of terminal	11.2.1.2.2.1.2	The bilateral agreement needs to address how the manufacturer provides their Client Root CA Certificate (or an intermediate) to the operator.
Visibility of installed OpApp code to users		The bilateral agreement may address whether the files of an installed OpApp (HTML, JavaScript, CSS, images, etc) are required to be invisible to users and, if so, how this is to be enforced.
Visibility of data of installed OpApp to users		The bilateral agreement may address whether data of an installed OpApp (cookies, Web Storage, private persistent storage, ...) is required to be invisible to users and, if so, how this is to be enforced.
Scope of operator application		

Topic	Reference	Description
Replacement and suppression of UI elements	6.4	The bilateral agreement needs to define which UI elements of the terminal are to be suppressed.the operator application replaces. • This may include the definition of additional (user defined) constants to be used with the `replaceUIElements` method. • This may include both UI elements marked as "M" in table A.2 and others. • This may include the definition of additional (user defined) constants to be used with the replaceUIElements method. Most UI elements of the terminal that are to be suppressed will be replaced by the operator application but the bilateral agreement may define exceptions that are suppressed without being replaced.
Suppression of UI elements	6.4	Subject to the bilateral agreement, terminals may support suppression of other groups of UI elements from that table and further UI elements defined by the manufacturer and operator (using values from the user defined range).
Additional UI elements that start an operator application	6.4	The bilateral agreement may define UI elements that, when selected by the user, cause an operator application to be started if it is not already running, e.g. pressing the PVR button.
Parental control	5.3.4.3.4 9.6	The bilateral agreement may define whether the operator application or the terminal implements the regulatory requirements for parental control. The bilateral agreement may also define additional use cases under which the terminal will preblock the audio and video services. The bilateral agreement may define that part of the parental control feature is implemented by the operator application and part by the terminal. If so then it needs to define which of the terminal or the operator application will show PIN code UIs and how they relate to each-other. The bilateral agreement may define that the parental control feature is implemented by the terminal for some channels (e.g. channels not offered by the operator – see clause 5.5.1) and by the operator application for others. The bilateral agreement may define an operator-specific or private parental control descriptor and that the terminal blocks channels with such a descriptor present until the operator application unblocks them.
Channels outside the set of channels aggregated by the operator	4.1.7.1.7 5.5.1	The bilateral agreement may define how the operator application or the terminal provide UI elements for overlaying channels outside the set of channels aggregated by the operator.
Key events	5.2.2 10.1.3	The bilateral agreement needs to define which of the operator application reserved keys and operator application keys are to be available to the operator application. • For an operator-specific operator application this may be extended with system keys and other keys. • It may also include details about the interaction devices or interaction methods which would generate these key events.
Scheduling recordings	A.2.1.4	The bilateral agreement may define that scheduled recordings can only be programmed via the operator application and not by the manufacturer UI.
Behaviour outside of the terminal's linear TV viewing mode	7.1.1 8.4.1.1	The bilateral agreement should address behaviour when the user selects an HDMI input or an app from a global VoD service provider or something similar that results in linear TV not being active. For example, The operator application may be stopped or it may keep running in background mode with "`opapp-silent`" context. Notifications may nor may not be shown.
Operator application presented channels	9.9	The bilateral agreement may define types or properties of channels that are to be considered operator application presented channels for which the operator application handles media presentation.

Topic	Reference	Description
Contextual menus for component selection and MSE	10.1.1.6	The bialateral agreement should address if clause 10.1.1.6 applies, i.e. if all the following apply; • The main window of the operator application presents video and audio using Media Source Extensions [21] and • The terminal has a UI for media component selection that adapts to the available set of media components and The operator application does not fully replace the terminal UI for media component selection
Terminal capabilities		
Channel scan	9.2	The bilateral agreement may define the channel scan requirements on the terminal. • This may include the implementation of the API to perform a channel scan. • This may refer to an existing operator specification used outside of the HbbTV® context.
Access to tuners and network resources during active standby	6.3.4	The bilaterial agreement may define whether the operator application has access to tuners and network resources during active standby.
Proprietary TV functions	8.3.1	If the terminal provides any proprietary TV functions to the operator application, the bilateral agreement should define these functions.
Performance		The bilateral agreement should address the performance that the terminal is expected to deliver when running the operator application. It may be easier and more helpful to define responsiveness of elements in the real application than trying to do this using benchmark programs. Examples could include the time taken from when the applications is started to either when the first JavaScript of the application is executed and/or to when the first graphical elements of the application are visible to the user.
Operator application video window	9.9.3	The bilateral agreement may define performance requirements and resource usage constraints that apply to the use of the operator application video window.
Use of API for operator application presented channels by the operator application main window	9.9.4	The bilateral agreement may specify if part or all of the API for operator application presented channels is made available to the operator application main window in addition to the operator application video window.
Product consistency		The bilateral agreement may include requirements to promote consistency and clarity of user experience across native terminal functions and operator application functions and between operator application presented channels, broadcast natively presented channels and broadband natively presented channels (e.g. in connection with any live pause features).
Version of TS 102 796 [1]		The bilateral agreement may require at least a specific version of TS 102 796 [1]. For example, operator application presented channels would typically rely on W3C Media Source Extensions and hence would require HbbTV 2.0.3 / TS 102 796 V1.6.1 or greater and may use preferredAudioLanguage47 and preferredSubtitleLanguage47 that are only found in HbbTV 2.0.4 / TS 102 796 V1.7.1.

Annex E (informative): Bilateral agreement profiles

E.1 Introduction

In this specification, a number of functionalities is left open for manufacturers and operators to decide if or how they would like to implement those features. To aid this process, Annex D was created to provide an overview on which topics may be included in such an agreement.

However, creating such a bilateral agreement has proven difficult due to the sheer number of possible combinations of features. Furthermore, due to the nature of bilateral agreements, this problem arises each time parties desire to negotiate an implementation and/or deployment of an Operator Application.

In order ease this process, this annex contains a number of profiles for Annex D, which may serve as an example or baseline for bilateral agreement negotiations. The profiles are structured such that the covered topics correspond to the structure of table D.1, and the headings in the profiles are (mostly) the same as those in annex D. Parties in the process of negotiating a bilateral agreement may decide to use (parts of) the following profiles as a tool to speed-up negotiations.

E.2 Profiles

E.2.1 IPTV Operator profile

E.2.1.1 Introduction

This IPTV operator profile aims to provide a framework of bilateral agreement topics for pure IP-based operator applications. In particular, the operator application is expected not to make use of the terminal's tuner system (if present). Where applicable, this profile attempts to provide reasonable defaults for certain topics and decisions. For topics that are controversial, this profile tries to specify the lowest common denominator in an attempt to establish a reasonable baseline for negotiations.

This profile considers two use-cases of IPTV operator applications:

1. The entire operator application is installed on the terminal. The operator application will fetch content and metadata over IP.
2. A small launcher operator application is installed on the terminal. When the launcher operator application is started, it will download a full implementation over IP and execute that implementation afterwards.

E.2.1.2 Quality assurance

The operator needs to document their operator application certification procedure. This includes text describing how they will test their operator application on the terminal.

1. The operator application needs to be tested on the terminal by the operator, or if applicable, by the operator's operator application supplier.

2. The operator needs to make the operator application available for testing by the manufacturer and, if applicable, by the supplier of the operator application implementation used by the manufacturer. The latest version of the operator application continues to be available after certification for regression testing.

3. Each party needs to nominate one responsible quality assurance manager for service development. Testing activities needs to cover the operator application as well as the terminal. The quality assurance manager should remain involved past the public launch of the operator application until a stable product and service is released.

E.2.1.3 Software updates

The terminal needs to support processing updates to an already installed operator application as defined in clause 6.2 of the present document.

When an operator wants to deploy an updated operator application (either on the terminal, or on a remote location which is loaded by a launcher operator application), that updated operator application needs to be tested on multiple receivers provided by the terminal manufacturer. To facilitate this, the following should apply:

1. Any issues need to be resolved in collaboration between the manufacturer and the operator.
2. The operator needs to make the updated operator application available to the manufacturer for regression testing.

When a manufacturer wants to deploy a software update to their terminals, the terminal with updated software needs to be tested on multiple operator application versions provided by the operator.

1. Any issues needs to be resolved with the manufacturer.
2. The manufacturer needs to provide the operator with (either local or remote) access to an updated terminal to allow regression testing.

For software updates that are not expected to cause regression, the operator and the manufacturer may define a lightweight test procedure. This lightweight procedure may be based on the above procedure.

NOTE: The process response time(s) may be agreed upon separately.

E.2.1.4 Branding

The operator is responsible for the branding of the operator application.

NOTE: An operator may control multiple brands, which may require publishing multiple operator applications. In this case, multiple brands will be visible during installation. It is the responsibility of the operator to provide users with appropriate documentation to guide them through an installation process where multiple operator applications from the same operator may be available.

The manufacturer needs to specify a required format and dimensions of any desired operator application icons, which the operator needs to provide. The list needs to include at least the following icon categories:

1. web notification icons,
2. icons used during the set-up process of the terminal,
3. if applicable, launcher icons on the home screen of the terminal,
4. icons used in the input source menu.

E.2.1.5 Organisation ID

The operator needs to provide a DVB `organisation_id` as defined in ETSI TS 102 809 [3].

The operator may change the number of available operator applications, the terminal needs to support installation of the operator applications which are available when the discovery process is triggered.

NOTE: The terminal installation UI may require a brand to be selected first, after which the terminal may only support installing a single associated operator application.

E.2.1.6 Size of the operator application

The terminal needs to support installing at least one operator application with an uncompressed size of at least 2MB.

NOTE: The above requirement is expected to be sufficient for launcher operator applications (see clause E.2.1.1). For non-launcher operator applications, the required storage capacity may be higher. In that case, the operator needs to propose a different value.

The terminal needs to support operator applications which require up to 150MB of RAM, this includes stack and heap memory, but excludes memory required by the media pipeline of the terminal.

E.2.1.7 Type of operator application

The terminal needs to support all required features for privileged operator applications.

E.2.1.8 Discovery

The terminal needs to support the discovery process of Table 5 (combination 2) of the present document, consisting of the following steps:

1. discovering the operator application's XML-AIT using a hardwired FQDN,
2. retrieving the discovered XML-AIT using an available broadband connection, and
3. retrieving the operator application package specified in said retrieved XML AIT using an available broadband connection.

The operator needs to provide a FQDN to be used for the discovery of an XML AIT.

The manufacturer needs to support using multiple FQDNs for a single operator. The note from E.2.1.5 may apply.

The operator may be implementing geo-blocking in their servers. If this is the case, the operator needs to indicate for each FQDN in which countries it is expected to be available.

Discovering the XML-AIT of an operator application requires the use of a DNS SRV service, according to clauses 6.1.3.5 and 6.1.4 of the present document. The operator needs to be responsible for providing this service and for ensuring its correct operation.

When the operator's service offering is only available on certain networks, users may end up installing an operator application which is unable to serve content. Therefore, the operator needs to ensure that the discovery for operator applications that are only functional on a limited set of networks will fail when the terminal is not connected to one of those networks.

E.2.1.9 Installation of the operator application

During the first-time set-up of the terminal:

1. The manufacturer needs to allow the installation of operator applications.
2. If applicable to the terminal's installation UI, the terminal needs to present operator applications as one of the supported input source options.

The terminal needs to support installing operator applications after its first-time installation from within the device UI. This may be necessary to install operator applications at a later time if during the first-time set-up of the terminal:

1. discovery fails, or
2. IP connectivity is temporarily unavailable,
3. the user directly or indirectly decides not to install the operator application (i.e. when the user decides to skip configuring IP connectivity).

E.2.1.10 Deciding which operator applications to install

The user needs to be able to decide which operator application(s) to install.

E.2.1.11 Failure of the installation of the operator application

When the installation of the operator application fails, the terminal needs to present minimal help to the user (i.e. a description of the error, steps for remediation of the error, and a reference to the device's FAQ) in its native look and feel.

E.2.1.12 Non-connected operation

The terminal is not expected to be able to install the operator application when no IP connectivity is available.

When the operator application is running and the terminal has access to IP connectivity, this connectivity is also available to the operator application.

When a temporary loss of IP connectivity occurs:

1. The terminal is not expected to notify the operator application.
2. The operator application needs to determine loss of connectivity by itself, e.g. by actively making periodic requests to a known server. This holds for all types of operator applications.
3. The operator application should provide a minimal set of functions for offline operations.

NOTE: IPTV operator applications rely on IP connectivity to implement their use-cases, but this cannot be guaranteed to always be available by the terminal. As a consequence, IPTV operator applications should be able to detect and deal with (temporary) disruptions in IP connectivity.

E.2.1.13 Use and size of the private persistent storage

The terminal needs to support the private persistent storage feature as described in the present document. The size of the private persistent storage needs to be at least 1MB per origin.

For launcher-based operator applications (see Annex E.2.1.1), the operator application may require more than a single origin (e.g. one for the origin of the launcher operator application, and another for the remote operator application).

E.2.1.14 Starting the operator application

After installation, the terminal needs to support starting an operator application in one of the following ways:

- Selecting the operator application from the input source menu (e.g. to allow users to switch back from HDMI to the operator application).
- When the user presses a dedicated IPTV button (e.g. the OPAPP key as defined in clause E.2.1.26 of the present document) on the remote control.
- If applicable, selecting the operator application from the home screen of the terminal.
- If applicable, selecting the operator application from other menus where the option to transition to HDMI input is available.

When the operator application was stopped as a result of launching a 3rd party application, and the user leaves that application, the terminal should restart the operator application.

When started, the operator application may return to the most recently watched content.

E.2.1.15 Stopping the operator application

When the terminal encounters an error from which it can recover, but which requires the operator application to be killed (e.g. when the operator application runs out of memory), the terminal needs to notify the user and attempts to restart the operator application at least once.

The terminal may attempt to detect repeated, similar fatal errors from an operator application and eventually no longer restart the app but show an error message instead.

In order to ensure the continued meaningful operation of the operator application, the terminal may restart the operator application when the operator application encounters an untrapped error (such as an uncaught exception).

In addition, the terminal needs to present meaningful error messages and context-sensitive help in the case one of the following events:

1. an error occurs when starting the operator application, or
2. an error occurs when updating the operator application.

The operator needs to ensure that a user cannot get trapped in the operator application UI. The user always needs to have a sequence of actions which they may perform to exit the operator application.

E.2.1.16 Operator application state change

No additional requirements.

E.2.1.17 Power-on behaviour

The following behaviour is based on expected regular terminal behaviour. It is not the intention of this clause to replace existing terminal behaviour.

When the terminal is turned on and has booted, depending on the input source state on shutdown, the terminal should behave as follows:

1. If the operator application was selected, it is started,
2. if the input source state is unknown, the input source is set to the operator application, and it is started,
3. else, the input source may be set to the state it had on shutdown.

E.2.1.18 Active standby

When entering the active standby power state:

1. the terminal is not required to keep the operator application running.
2. (consequentially) the terminal is not required to provide tuner or network resources to the operator application.

The terminal may support suspending the operator application when entering the active or passive standby power states. If this is not the case, the operator application needs to be killed according to clause 6.3.4. Operator applications are encouraged to use the `window.onBeforeUnload` event to respond to being killed (e.g. to save their current state to a remote service).

NOTE: Some terminals do not support the active standby state.

E.2.1.19 Start pages of the operator application

The operator application may use cookies or persistent storage to determine the desired application-specific startup state from the start page.

E.2.1.20 Visible RF-Based channel list

This feature is not applicable for IPTV operator applications. IPTV operator applications are expected to use an in-app channel list not based on any input by the terminal's tuner system (if present).

E.2.1.21 Mandatory API

No additional requirements.

E.2.1.22 Web notifications

No additional requirements. The terminal needs to support web notifications as mandated by the present document.

E.2.1.23 Operator application state change

No additional requirements.

E.2.1.24 Error codes

The manufacturer needs to define a list of errors and corresponding error codes used for reporting errors during updates according to clause A.2.2.2 of the present document. These error codes are categorized in said annex as follows:

1. Errors during the downloading of updated software

2. Errors during the installation of updated software

E.2.1.25 Authentication, encryption and decryption of the operator application

Activities and procedures related to authentication and encryption may depend on internal business processes of the operator. The following is recommended based on clause 11 of the present document:

The operator needs to an Operator Signing Certificate and Operator Signing Root CA or Operator Signing Intermediary CA certificates to the manufacturer (see clause 11.3.2 of the present document).

The manufacturer needs to a Terminal Packaging Certificate and Manufacturer Packaging Root CA or Terminal Packaging Intermediate CA certificates to the operator (see clause 11.3.3 of the present document).

The operator needs to and encrypt the operator application package using the above certificate chains according to clause 11.3.4 of the present document.

The manufacturer needs to use the above certificate chains to decrypt and validate the operator application package according to clause 11.3.4.5 of the present document.

E.2.1.26 Code and data visiblity

The operator application code needs to be invisible to the user. The operator and the manufacturer need to discuss how this is implemented on the terminal.

The terminal does not need to take any additional measures to hide operator application data.

E.2.1.27 Replacement and suppression of UI elements

The terminal needs to simultaneously support replacing the following UI elements (if the terminal supports these elements):

1. UI_TVMODE
2. UI_EPG

If the terminal supports the following UI elements, the terminal also needs to support suppression of each:

1. UI_TIMESHIFT
2. UI_RECORD
3. UI_PVR

The operator needs to ensure that when the operator application makes a call to the `replaceUIElements` method (see Annex A.2.1.3), the `elements` argument is an array containing all and only those constants for the (supported) UI elements mentioned above.

The terminal may support replacing all subsets of the above constants.

NOTE: A number of IPTV operators may desire to replace the UI elements corresponding to the UI_VOLUME constant. In such cases, the bilateral agreement with such operators needs to be extended to include support for that constant.

E.2.1.28 Key events

The terminal needs to support replacement of the regular functionality and provide key events for the following keys (if the device supports those keys):

1. VK_CHANNEL_UP
2. VK_CHANNEL_DOWN
3. VK_INFO
4. VK_GUIDE
5. VK_CHANNELS
6. VK_AUDIO_TRACK
7. VK_AUDIO_DESC

8. VK_SUBTITLE
9. VK_STOP
10. VK_PLAY
11. VK_PAUSE
12. VK_PLAY_PAUSE
13. VK_FAST_FWD
14. VK_REWIND
15. VK_RECORD

Additionally, the terminal needs to support providing key events for all regular HbbTV application keys as listed by Table 17. The operator and manufacturer need to designate a particular key as the OPAPP key. Unless agreed otherwise, the VK_GUIDE key needs to serve this purpose.

When the terminal is in standby mode, it needs to respond to the OPAPP key by:

1. moving to the power-on state, and
2. launching the operator application.

The operator application needs to register event handlers for all keys which are required to be supported by the terminal according to the above.

NOTE: Terminals may not support mixed operation scenarios where only a subset of the keys required to be supported by the terminal according to the above are handled by the operator application and some by the terminal implementation.

If the terminal supports replacing UI elements corresponding to the UI_VOLUME constant, the terminal needs to support replacement of the regular functionality and provide key events for the following additional keys:

1. VK_VOLUME_UP
2. VK_VOLUME_DOWN
3. VK_MUTE

Operator applications that request a key always needs to exhibit a visible response when that key is pressed in order to prevent the user encountering unresponsive (e.g. 'dead') keys.

Some terminals may support more keys than defined in this bilateral agreement. The operator application is not expected to request such keys.

E.2.1.29 Parental control

The operator application is responsible for implementing parental control.

E.2.1.30 Scheduling recordings

Not applicable for IPTV operator applications. The operator application may implement a proprietary form of scheduled recordings (e.g. cloud-based recordings) as part of the application.

E.2.1.31 Channel scan

Not applicable for IPTV operator applications. The operator application is esponsible for providing and maintaining its own channel list. The operator application may retrieve a channel list over broadband.

E.2.1.32 Proprietary TV functions

The terminal may support launching the operator application from a companion device using an appropriate zero-configuration protocol (such as the DIAL protocol [i.7]).

The operator application needs to be able to use custom DASH players, therefore the terminal needs to support Media Source Extensions [21] or a functionally equivalent API.

Additionally, the operator application needs to be able to gather playback quality metrics, therefore the terminal needs to support the Media Playback Quality API [22] or a functionally equivalent API.

The operator needs to define the required DRM systems, their minimum versions and how they need to be configured.

E.2.1.33 Performance

The operator needs to define expected values for:

1. The time taken from when the user starts the operator application until the first line of JavaScript code is executed.
2. The time taken from when the user starts the operator application until the internet connection becomes available to the operator application.
3. The time taken from when the operator application is started until the first graphical elements of the operator application become visible to the user.

The requirements below refer to the following concept from the CTA-WAVE 206 specification [23]:

– Startup time, time from when the user intends for playback to start to when playback has perceptibly begun. This is defined by [23] as the 'initialStartupTime' metric.

The following performance requirements are based on native application performance at the time of writing. The terminal should meet these for DRM encrypted media which has been optimized for meeting these requirements.

1. After the user has pressed the relevant button on their remote, the terminal should support changing to a different (IPTV) channel and display content within 1.5 seconds.
2. Given A/V content, which is correctly synchronized, after smooth playback has been achieved, the terminal should support A/V synchronisation where audio content is played at most 45ms before and at most 120ms after the video content.
3. When the user starts the terminal from standby mode (e.g. after the user presses the power button), the terminal should support a startup time for linear content within 10 seconds.
4. If adaptive streaming is used, given sufficient available bandwidth and a suitable DASH player implementation, when streaming is started, the terminal should support selecting the highest available bandwidth within 5 seconds.

Annex F (normative): Mapping between HTML5 video element and CICAM player mode

F.1 Introduction (informative)

This annex defines how an HbbTV terminal is able to delegate requests to play content that originate from an HbbTV application to a media player in a CICAM. Specifically it defines the integration between the video element API from HTML5 [19] and the "Host-initiated playback mode" of the CICAM player resource from ETSI TS 103 205 [7].

NOTE 1: This annex only applies to the HTML5 video element and not to the HTML5 audio element.

Behaviour of either the HTML5 video element or the HbbTV terminal implementation of the CI+ protocol that is not related to the integration between them is outside the scope of this annex.

When the CICAM player resource is used in "Host-initiated playback mode":

- a media player in the CICAM is used to present content controlled by the HTML5 video element instead of the media player in the HbbTV terminal that is normally used;
- the HbbTV application controls media playback through the methods and properties of the HTML5 video element API which are then translated by the HbbTV terminal to the messages (called "APDU"s) supported by the CICAM player resource;
- the media player in the CICAM handles IP delivery protocols in place of the HbbTV terminal, and returns the content to HbbTV terminal as a SPTS.

Tables F.1 and F.2 summarize this integration firstly from the viewpoint of the CICAM (by listing the APDUs defined for the CICAM player resource) and secondly from the point of view of the HTML page (by listing the HTML5 video element methods and attributes).

Table F.1: CICAM Player resource APDUs

APDU	Direction	APDU usage	Reference
CICAM_player_verify_req	Host -> CICAM	Requests CICAM to check to see if the content can be consumed.	F.3
CICAM_player_verify_reply	CICAM -> Host	Signals if it is able to consume the content.	F.3
CICAM_player_capabilities_req	CICAM -> Host	Enables the CICAM to acquire codec capabilities from the Host.	Outside the scope of this annex.
CICAM_player_capabilities_reply	Host -> CICAM	The set of codecs that the Host supports.	Outside the scope of this annex.
CICAM_player_start_req	CICAM -> Host	Request to start player session	F.3
CICAM_player_start_reply	Host -> CICAM	Response to start request	F.3
CICAM_player_play_req	Host -> CICAM	Request by host to play a content item.	F.3
CICAM_player_status_error	CICAM -> Host	Signal critical error to Host	F.7
CICAM_player_control_req	Host -> CICAM	Instructs CICAM to seek to a given position and/or change playback speed.	F.8, F.9
CICAM_player_info_req	Host -> CICAM	Request for play speed, position, and duration	F.4, F.8, F.9
CICAM_player_info_reply	CICAM -> Host	Play speed, position and duration	F.4, F.8, F.9
CICAM_player_stop	Host -> CICAM	Instruct CICAM to terminate playback	F.4, F.5
CICAM_player_end	CICAM -> Host	Free playback session on Host	F.5
CICAM_player_asset_end	CICAM -> Host	Informs Host that end of asset has been reached	F.6
CICAM_player_update_req	CICAM -> Host	CICAM requests to provide updated components	F.10
CICAM_player_update_reply	Host -> CICAM	Host responds to player update request	Outside the scope of this annex.

Table F.2: HTML5 video element attributes and methods

Name	Description	Reference
readonly attribute MediaError? Error	Last error in media playback - one of aborted, network error, decode error, content not supported.	F.7
attribute DOMString src	URL of content item, can be written to set a new content item.	F.2, F.5
readonly attribute DOMString currentSrc	URL of current content item or empty string.	F.2
attribute DOMString crossOrigin	Use of cookies (etc) for cross-origin requests.	F.11
readonly attribute unsigned short networkState	One of not initialized, initialized but not using network, loading data, initializing.	F.4
attribute DOMString preload	Hint of extent to which content item may be preloaded (none, metadata only, optimistic download appropriate).	Outside the scope of this annex.
readonly attribute TimeRanges buffered;	Returns the part of the content item that is buffered locally.	F.4
void load()	Reload the media element after changing the source or other properties.	F.3, F.5
CanPlayTypeEnum canPlayType(DOMString type)	Test if the MIME type is one that can be decoded, cannot be decoded or cannot be determined.	F.11
readonly attribute unsigned short readyState	Return how much of the metadata & data for the content item has been loaded - nothing, just metadata, some data but not enough to start playing, enough data to start playing.	F.4
readonly attribute boolean seeking	Indicates that an asynchronous seek is in progress.	F.9
attribute double currentTime	The current play position. Writing to this starts a seek.	F.4, F.9
readonly attribute unrestricted double duration	The time of the end of the content item.	F.4, F.9
Date getStartDate()	An explicit date and time corresponding to the zero time in the media timeline.	F.11
readonly attribute boolean paused	Indicates whether playback is paused.	F.8
attribute double defaultPlaybackRate	Default playback speed ("The defaultPlaybackRate is used by the user agent when it exposes a user interface to the user.")	Outside the scope of this annex.
attribute double playbackRate	On reading, returns the current playback rate. On writing, attempts to change the playback rate.	F.8
readonly attribute TimeRanges played	represents the ranges of points on the media timeline of the media resource reached through the usual monotonic increase of the current playback position during normal playback.	F.4
readonly attribute TimeRanges seekable	represents the ranges of the media resource, if any, that the user agent is able to seek to, at the time the attribute is evaluated.	F.4, F.9
readonly attribute boolean ended	Indicates that playback reached the end of content.	F.6
attribute boolean autoplay	Automatically begin playback of the media resource as soon as enough data is available to do so without stopping/pausing.	Outside the scope of this annex.
attribute boolean loop	If set, automatically seek back to the start of the media resource upon reaching the end and play from there.	Outside the scope of this annex.
void play()	Play the content.	F.3, F.5
void pause()	Pause the content presentation.	F.8
attribute boolean controls	Indicates whether the HTML page has provided a UI for control of media playback otherwise the user agent should do this.	Outside the scope of this annex.
attribute double volume	Audio playback volume between 0.0 and 1.0.	Outside the scope of this annex.
attribute boolean muted	On reading, indicate if audio is muted. On writing mute or unmute audio.	Outside the scope of this annex.
attribute boolean defaultMuted	Indicate that audio should default to muted when presentation starts.	Outside the scope of this annex.
readonly attribute AudioTrackList audioTracks	List of audio tracks in the content item.	F.10
readonly attribute VideoTrackList videoTracks	List of video tracks in the content item.	F.10
readonly attribute TextTrackList textTracks	List of subtitle and other tracks in the content item.	F.10

Name	Description	Reference
TextTrack addTextTrack(TextTrackKind kind, optional DOMString label, optional DOMString language)	Creates and returns a new TextTrack object, which is also added to the media element's list of text tracks.	F.11

NOTE 2: If the communication interface between HbbTV application and CICAM defined in this annex is not rich enough then, subject to support by the CICAM, the message passing mechanism between the 'application/oipfDrmAgent' embedded object and the CI Plus protocol defined in clause 11.4.1 of [1] can be used in addition.

F.2 Determining when to use CICAM player mode

When a Content Access Streaming Descriptor including the `CICAMPlayerPreferred` element is used with an HTML5 `video` element then the HbbTV terminal shall attempt to present the content referenced by the `contentURL` element using CICAM player mode. The HbbTV terminal shall not attempt to use CICAM player mode under other circumstances - e.g. A/V control object, HTML5 video element with `src` or `source` directly referencing content, HTML5 video element with `src` or `source` referencing Content Access Streaming Descriptor without `CICAMPlayerPreferred` element or HTML5 video element with `src` or `source` referencing DASH MPD.

F.3 Starting CICAM player

The process for session initialization of Host-Initiated playback as described in clause 8.3.2 of ETSI TS 103 205 [7] shall apply for an HTML5 video element where it has been determined that a CICAM player will be used (according to clause F.2) and either the `load()` or `play()` methods are called or the `autoplay` property is set and takes effect. The URL from the `<ContentURL>` element shall be wrapped in a ServiceLocation and passed to the CICAM in the `CICAM_player_play_req()` and/or `CICAM_player_verify_req()` APDUs. When available in the Content Access Streaming Descriptor, the `DRMControlInformation` element obtained from the Content Access Streaming Descriptor shall also be wrapped in the ServiceLocation passed to the CICAM. The ServiceLocation shall have no ContentAttributes and the priority attribute shall be set to zero.

F.4 During CICAM player use

With reference to the sequence diagram for host-initiated playback in clause 8.7 of ETSI TS 103 205 [7], the `networkState` attribute shall be updated as defined by the resource selection and resource fetch algorithms in the HTML5 specification except as follows:

- When the `input_max_bitrate` requested by the CICAM is not zero:
  - The value `NETWORK_IDLE` shall be returned between when the HbbTV terminal has received the `CICAM_player_start_req()` APDU from the CICAM and when the HbbTV terminal sends the `comms_info_reply()` APDU to the CICAM.
  - The value `NETWORK_LOADING` shall be returned between when the HbbTV terminal sends the `comms_info_reply()` APDU to the CICAM until the HbbTV terminal sends the `CICAM_player_stop()` APDU to the CICAM.
- When the `input_max_bitrate` requested by the CICAM is zero (CICAM uses its own connectivity):
  - The value `NETWORK_LOADING` shall be returned between when the HbbTV terminal has received the `CICAM_player_start_req()` APDU from the CICAM until the HbbTV terminal sends the `CICAM_player_stop()` APDU to the CICAM.

The value of the `readyState` attribute shall as follows:

- `HAVE_NOTHING` when the `networkState` attribute has values other than `NETWORK_LOADING`.

- `HAVE_ENOUGH_DATA` when the `networkState` attribute has the value `NETWORK_LOADING`.

The `buffered` attribute shall return a `TimeRanges` object as follows:

- If the duration of the content is known (i.e. the `duration` field in the last `CICAM_player_info_reply()` APDU was not 0xFFFFFFFF) then the `TimeRanges` object shall represent the full duration of the content item as defined for the value of the `duration` attribute.
- If the duration of the content is not known (i.e. the `duration` field in the last `CICAM_player_info_reply()` APDU was 0xFFFFFFFF) then the `TimeRanges` object shall represent 5 seconds either side of the current play position.

The `played` attribute shall be calculated by the HbbTV terminal as defined in the HTML5 specification using the values of the current playback position.

An HbbTV application reading the current playback position (e.g. `currentTime` property) shall be handled as follows:

- HbbTV terminals shall maintain a local copy of the current playback position as defined in the HTML5 specification [19] and update this as playback proceeds and return this to applications.
- HbbTV terminals shall poll the CICAM player current play position by sending the `CICAM_player_info_req()` APDU. This shall be done at intervals of not more than 30 seconds.
- If the resulting `CICAM_player_info_reply()` APDU from the CICAM includes a position that is not 0xFFFFFFFF then the terminal shall re-synchronize the local copy of the current playback position from this value and then update the value as playback proceeds until the next poll.
- HbbTV terminals shall not send a `CICAM_player_info_req()` APDU each time an application reads the current playback position and then block until a `CICAM_player_info_reply()` APDU is received.

The HbbTV terminal shall maintain a local copy of the duration of the content and return this when the duration attribute is read. Each time a `CICAM_player_info_reply()` APDU is received from the CICAM, if the value of the duration field in that APDU is not 0xFFFFFFFF then the local copy of the duration shall be updated if different. Each time the local copy is updated then a `TimeRanges` object shall be generated to be returned from the `seekable` property which shall contain a single range whose start is zero and whose end is the duration returned from the CICAM.

If the content item referenced from the video element is changed to a different one and the conditions in clause F.2 above apply for the updated content item, then the CICAM player session shall be stopped and a new CICAM player session shall be started as defined in clause F.3.

F.5 Stopping CICAM player use

An HTML5 video element shall cease using a CICAM player, according to clauses 8.3.2 and 8.6.3 of ETSI TS 103 205 [7], if any of the following apply:

- The content item referenced from the video element is changed to a different one such that none of the conditions in clause F.2 apply to the new content item and either the `load()` or `play()` methods are called.
- The `src` attribute of the `video` element is set to empty and the `load()` method is called.
- The HTML5 video element has released the video and audio decoder resources as defined in clause 9.6.2 of [1].
- The HTML5 video element is garbage collected following removal from the DOM tree.

F.6 Play to end of content

If the CICAM player sends a `CICAM_player_asset_end()` APDU to the HbbTV terminal with the `beginning` field set to '0' then the procedure for "When the current playback position reaches the end of the media resource when the direction of playback is forwards" shall be followed as defined in clause 4.7.10.8 of the HTML5 specification [19].

If the CICAM player sends a `CICAM_player_asset_end()` APDU to the HbbTV terminal with the `beginning` field set to '1' then the procedure for "When the current playback position reaches the earliest possible position of the media resource when the direction of playback is backwards" shall be followed as defined in clause 4.7.10.8 of the HTML5 specification [19].

F.7 Errors

When the HbbTV terminal receives a `CICAM_player_status_error()` APDU from the CICAM, it shall set the `error` attribute with the mapping shown in Table F.3.

Table F.3: Mapping of error values

player_status from the CICAM		MediaError? Error	
0x01	Error - content play is not possible (e.g. unsupported content format or protocol)	4	`MEDIA_ERR_SRC_NOT_SUPPORTED`
0x02	Error - unrecoverable error	4	`MEDIA_ERR_SRC_NOT_SUPPORTED`
0x03	Error - content blocked (e.g. no content license available)	3	`MEDIA_ERR_DECODE`

Since the HbbTV terminal has no information about the CICAM buffering or IP connectivity, IP connectivity issues and errors in the HbbTV terminal shall not be directly reported as errors via the HTML5 video element.

F.8 Play speed

An application setting the playbackRate property shall result in the terminal sending a `CICAM_player_control_req()` APDU to the CICAM with the `command` element set to 0x02 and the `Speed` element being the value set by the application converted from decimal to "hundredths of the nominal speed".

An application calling the `pause()` method shall result in the terminal sending a `CICAM_player_control_req()` APDU to the CICAM with the `command` element set to 0x02 and the `Speed` element being forced to 0.

The terminal shall periodically query the CICAM player current speed by sending the `CICAM_player_info_req()` APDU. The terminal shall maintain a cached copy of the CICAM player speed returned in the `CICAM_player_info_reply()` APDU.

An application reading the `playbackRate` property shall result in the terminal returning the cached CICAM player speed divided by 100.

An application reading the `paused` property shall result in the terminal returning `true` if the cached CICAM player speed is zero. Otherwise the terminal returns `false`.

F.9 Random access

In the seeking procedure defined in clause 4.7.10.9 of the HTML5 specification [19], immediately following step #11, "Set the current playback position to the given new playback position", the terminal shall send a `CICAM_player_control_req()` APDU to the CICAM with the `command` field set to 0x01, the `seek_mode` field set to 0x00 and the seek position being the newly set current playback position expressed in milliseconds.

Only when the CICAM replies with a `CICAM_player_info_reply()` APDU shall the seeking procedure resume with step #12 "Wait until the user agent has established whether or not the media data for the new playback position is available, and, if it is, until it has decoded enough data to play back that position."

F.10 Tracks

`VideoTrack`, `AudioTrack` and `TextTrack` objects shall be created for content delivered by a CICAM player as defined in clause A.2.12.1 of [1] for MPEG-2 transport streams delivered to the HbbTV terminal by other mechanisms. Changes in the elementary streams delivered from the CICAM (i.e. the CICAM sending a `CICAM_player_update_req()` APDU to the host) shall result in new track objects being created based on the information in the new PMT delivered by the CICAM.

NOTE: When more than one stream of the same media type is available (e.g. audio tracks in multiple languages or for accessibility), the CICAM is expected to deliver all available streams to the HbbTV terminal.

The mechanisms for component selection defined in clause 10.2.7 of [1] shall be supported for the stream delivered from the CICAM in the same way as they would be supported for an MPEG-2 transport stream streamed over HTTP.

F.11 No mapping possible

No mapping of the following methods or properties from the HTML5 video element is possible:

- The `crossOrigin` attribute will be ignored by the CICAM player.
- The `getStartDate()` method shall return a `Date` object representing "NaN".
- The `canPlayType` method shall return the same result for a given input regardless of the presence or absence of a CICAM supporting CICAM player mode.
- HbbTV applications should not call the `addTextTrack()` method for a video element being presented by a CICAM player. Any track objects created shall have the readiness state set to 'Failed to load'.

Annex G (informative): Sequence diagrams for parental access control

G.1 Operator Application Triggered Channel Selection for Operator Offered Channels

Figure G.1G.1 shows the process of channel selection triggered by an operator application for channels offered by that operator.

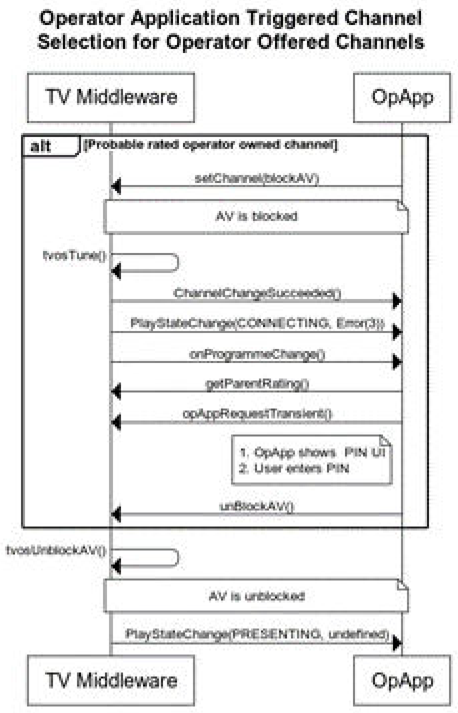

Figure G.1: Operator Application Triggered Channel Selection for Operator Offered Channels

G.2 Operator Application triggered Channel Change non operator offered channel

Figure G.2G.2 shows the process of channel selection triggered by an operator application for channels not offered by that operator (see clause 5.5.1).

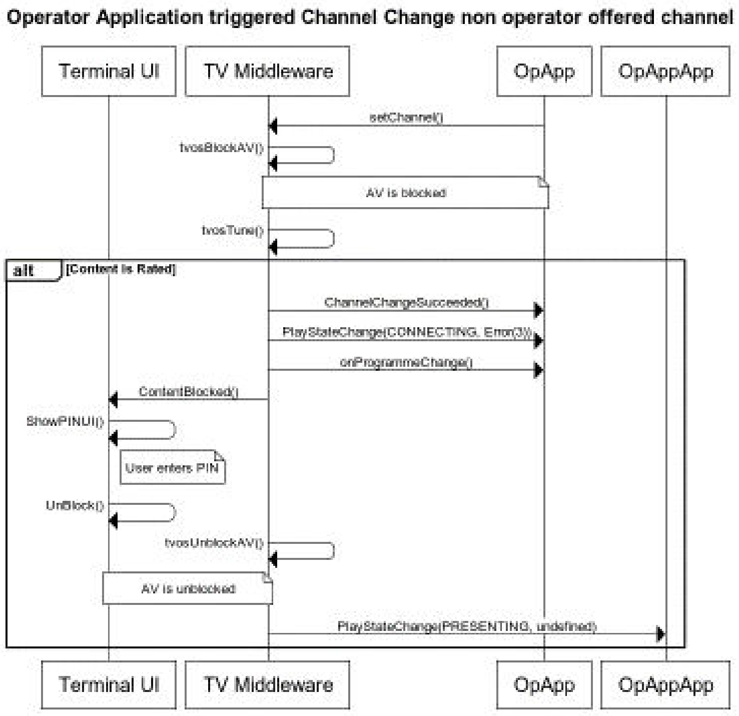

Figure G.2: Operator Application triggered Channel Change for non-operator offered channel

G.3 Terminal Triggered Channel Change for operator offered Channel

Figure G.3 shows the process of channel selection triggered by the terminal for channels offered by the operator.

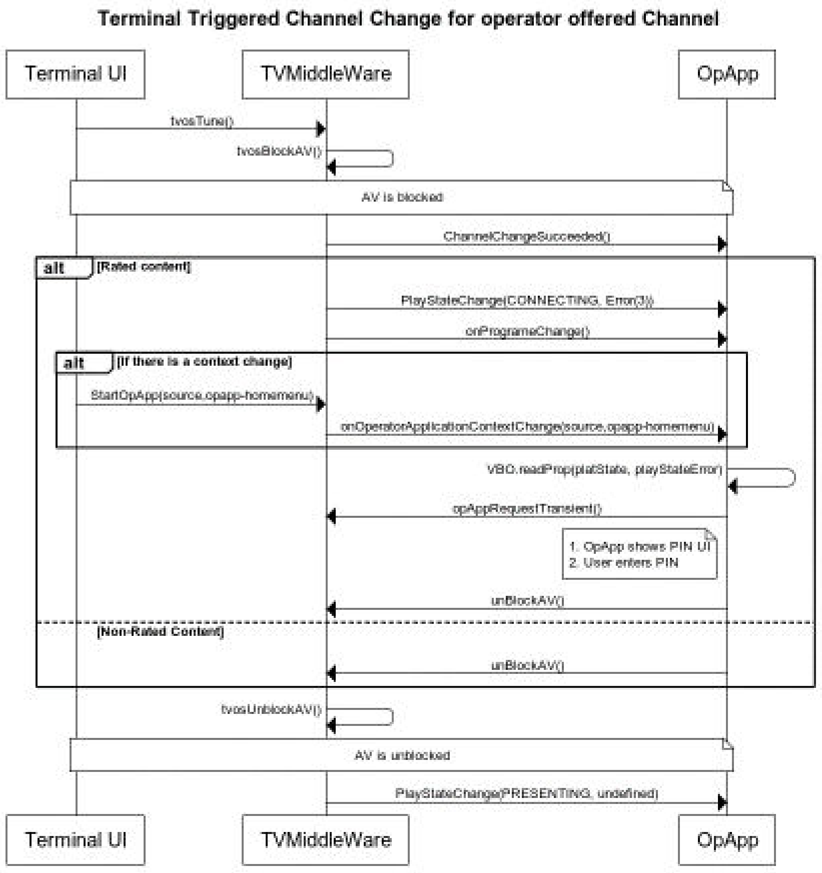
Terminal Triggered Channel Change for operator offered Channel
Terminal UI
TVMiddleWare
OpApp
tvosTune()
tvosBlockAV()
AV is blocked
ChannelChangeSucceeded()
alt
[Rated content]
PlayStateChange(CONNECTING, Error(3))
onProgrameChange()
alt
[If there is a context change]
StartOpApp(source,opapp-homemenu)
onOperatorApplicationContextChange(source,opapp-homemenu)
VBO.readProp(platState, playStateError)
opAppRequestTransient()
1. OpApp shows PIN UI
2. User enters PIN
unBlockAV()
[Non-Rated Content]
unBlockAV()
tvosUnblockAV()
AV is unblocked
PlayStateChange(PRESENTING, undefined)
Terminal UI
TVMiddleWare
OpApp

Figure G.3: Terminal Triggered Channel Change for operator offered Channel

G.4 Terminal Triggers non Operator Offered Channel Change

Figure G.4 shows the process of channel selection triggered by the terminal for channels not offered by the operator (see clause 5.5.1).

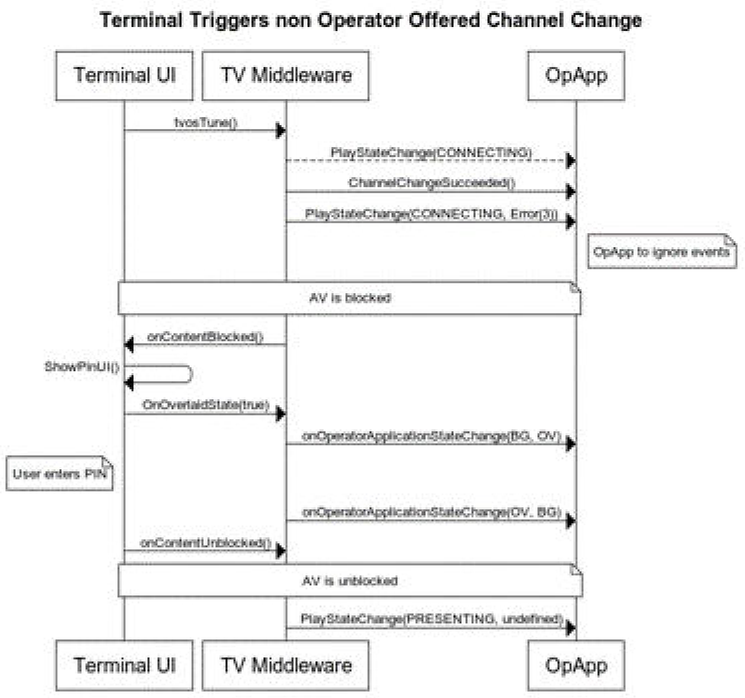
Terminal Triggers non Operator Offered Channel Change
Terminal UI
TV Middleware
OpApp
tvosTune()
PlayStateChange(CONNECTING)
ChannelChangeSucceeded()
PlayStateChange(CONNECTING, Error(3))
OpApp to ignore events
AV is blocked
onContentBlocked()
ShowPinUI()
OnOverlaidState(true)
onOperatorApplicationStateChange(BG, OV)
User enters PIN
onOperatorApplicationStateChange(OV, BG)
onContentUnblocked()
AV is unblocked
PlayStateChange(PRESENTING, undefined)
Terminal UI
TV Middleware
OpApp

Figure G.4: Terminal Triggers non Operator Offered Channel Change

G.5 Program Change on OpApp Offered Rated Channel

Figure G.5 shows the process that happens at the start of a programme carrying a parental access rating on a channel offered by the operator.

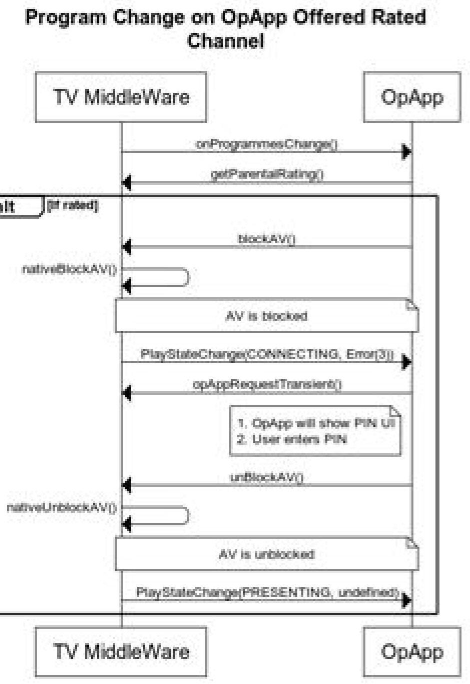

Figure G.5: Program Change on OpApp Offered Rated Channel

G.6 Channel Selection for Locked, non Rated Channel

Figure G.6 shows the process of channel selection for a channel that is manually blocked and does not have any associated parental rating signalling.

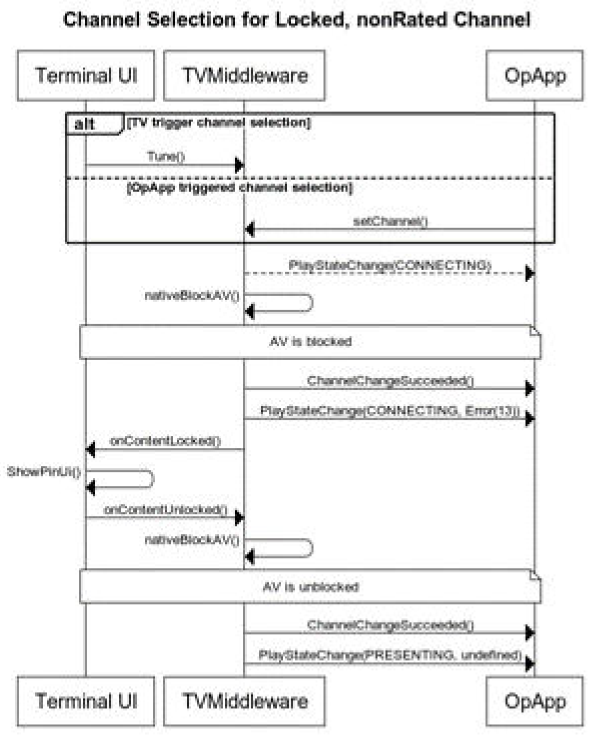

Figure G.6: Channel Selection for Locked, non-rated Channel

Annex H (normative): Electronic attachments

The present document includes an electronic attachment ts_103606v010201p0.zip with the following contents:

- response__org.hbbtv.ipplay.schema.json
  This is the normative JSON schema that will validate a correctly formed JSON-RPC response by an operator application to a terminal and by a terminal to an operator application, as referred to in clause 9.9.4.2.3 of the present document.

- request__org.hbbtv.ipplayback.mediaPositionUpdate.schema.json
  This is a normative JSON schema that will validate a correctly formed JSON-RPC request with method name `org.hbbtv.ipplayback.mediaPositionUpdate`, as referred to in clause 9.9.4.4.2 of the present document.

- request__org.hbbtv.ipplayback.setComponents.schema.json
  This is a normative JSON schema that will validate a correctly formed JSON-RPC request with method name `org.hbbtv.ipplayback.setComponents`, as referred to in clause 9.9.4.4.3 of the present document.

- request__org.hbbtv.ipplayback.setPresentFollowing.schema.json
  This is a normative JSON schema that will validate a correctly formed JSON-RPC request with method name `org.hbbtv.ipplayback.setPresentFollowing`, as referred to in clause 9.9.4.4.5 of the present document.

- request__org.hbbtv.ipplayback.setTimelineMapping.schema.json
  This is a normative JSON schema that will validate a correctly formed JSON-RPC request with method name `org.hbbtv.ipplayback.setTimelineMapping`, as referred to in clause 9.9.4.4.4 of the present document.

- request__org.hbbtv.ipplayback.statusUpdate.schema.json
  This is a normative JSON schema that will validate a correctly formed JSON-RPC request with method name `org.hbbtv.ipplayback.statusUpdate`, as referred to in clause 9.9.4.4.1 of the present document.

- request__org.hbbtv.ipplayer.pause.schema.json
  This is a normative JSON schema that will validate a correctly formed JSON-RPC request with method name `org.hbbtv.ipplayer.pause.schema`, as referred to in clause 9.9.4.5.6 of the present document.

- request__org.hbbtv.ipplayer.play.schema.json
  This is a normative JSON schema that will validate a correctly formed JSON-RPC request with method name `org.hbbtv.ipplayer.play.schema`, as referred to in clause 9.9.4.5.3 of the present document.

- request__org.hbbtv.ipplayer.resolveTimeline.schema.json
  This is a normative JSON schema that will validate a correctly formed JSON-RPC request with method name `org.hbbtv.ipplayer.resolveTimeline`, as referred to in clause 9.9.4.5.10 of the present document.

- request__org.hbbtv.ipplayer.resume.schema.json
  This is a normative JSON schema that will validate a correctly formed JSON-RPC request with method name `org.hbbtv.ipplayer.resume`, as referred to in clause 9.9.4.5.7 of the present document.

- request__org.hbbtv.ipplayer.seek.schema.json
  This is a normative JSON schema that will validate a correctly formed JSON-RPC request with method name `org.hbbtv.ipplayer.seek`, as referred to in clause 9.9.4.5.8 of the present document.

- request__org.hbbtv.ipplayer.selectChannel.schema.json
  This is a normative JSON schema that will validate a correctly formed JSON-RPC request with method name `org.hbbtv.ipplayer.selectChannel`, as referred to in clause 9.9.4.5.1 of the present document.

- response__org.hbbtv.ipplayer.selectChannel.schema.json
  This is a normative JSON schema that will validate a correctly formed JSON-RPC response to a request with method name `org.hbbtv.ipplayer.selectChannel`, as referred to in clause 9.9.4.5.1 of the present document.

- request__org.hbbtv.ipplayer.selectComponents.schema.json
  This is a normative JSON schema that will validate a correctly formed JSON-RPC request with method name `org.hbbtv.ipplayer.selectComponents`, as referred to in clause 9.9.4.5.9 of the present document.

- request__org.hbbtv.ipplayer.setVideoWindow.schema.json
  This is a normative JSON schema that will validate a correctly formed JSON-RPC request with method name `org.hbbtv.ipplayer.setVideoWindow`, as referred to in clause 9.9.4.5.4 of the present document.

- request__org.hbbtv.ipplayer.setRelativeVolume.schema.json
  This is a normative JSON schema that will validate a correctly formed JSON-RPC request with method name `org.hbbtv.ipplayer.setRelativeVolume`, as referred to in clause 9.9.4.5.5 of the present document.

- request__org.hbbtv.ipplayer.stop.schema.json
  This is a normative JSON schema that will validate a correctly formed JSON-RPC request with method name `org.hbbtv.ipplayer.stop`, as referred to in clause 9.9.4.5.2 of the present document.

History

Document history		
V1.1.1	May 2018	Publication