Operator Applications

2017-12-15

Contents

1 Scope

The present document specifies a platform, based on the existing HbbTV specification 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 main characteristics of the platform with respect to the execution of operator applications are the following:

- Available operator applications can be discovered and authenticated.
- Operator applications from different providers may be accessible from the same terminal.
- Operator applications can be delivered via broadcast or broadband.
- Standardized APIs allow the deployment of operator applications across different terminal manufacturers.
- Access to an HbbTV terminal is controllable on a model by model basis.
- Operator applications can coexist with regular HbbTV applications.
- Operator applications can communicate with a companion device.

The present document makes use of functionalities described in TS 102 796 [1] 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 "IP-delivered Broadcast Channels and Related Signalling of HbbTV Applications" [i.4].

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 is intended to be usable without additional country/market-specific specifications. It is however also possible to combine it with country/market-specific specifications.

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.

Referenced documents which are not found to be publicly available in the expected location might be found at https://docbox.etsi.org/Reference.

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/notifications/

[9] IETF, RFC1951: “DEFLATE Compressed Data Format Specification version 1.3”

NOTE: Available at: https://tools.ietf.org/html/rfc1951

[10] 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

[11] APPNOTE, ZIP File Format Specification, PKWARE

NOTE: Available at: http://www.pkware.com/documents/APPNOTE/APPNOTE-6.3.3.TXT

[12] IETF, RFC 5246: “The Transport Layer Security (TLS) Protocol Version 1.2”,

NOTE: Available at http://www.ietf.org/rfc/rfc5246.txt

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

NOTE: Available at http://www.ietf.org/rfc/rfc5280.txt

[14] IETF RFC 5652: “Cryptographic Message Syntax (CMS)”

NOTE: Available at http://www.ietf.org/rfc/rfc5652.txt

[15] 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”

NOTE: Available at http://www.ietf.org/rfc/rfc4055.txt

[16] ETSI EN 300 468: “Digital Video Broadcasting (DVB); Specification for Service Information (SI) in DVB systems”

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

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

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

NOTE: Available at http://www.ietf.org/rfc/rfc3986.txt

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”

3 Definitions, symbols 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-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: an HbbTV application that uses the features defined in TS 102 796 [1] and nothing from the present document except for features specifically identified as being available to regular HbbTV applications

NOTE: An example of a feature in the present document that is specifically identified as being available to regular HbbTV applications is `Configuration.runningOperatorApplication`.

standard operator application: application providing operator functionality using only the features defined in TS 102 796 [1]

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

terminal: HbbTV® terminal (as defined in ETSI TS 102 796 [1]) 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 Programe 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
CSS	Cascading Style Sheets
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
DTT	Digital Terrestrial Television
DVB	Digital Video Broadcasting
DVB-SI	Digital Video Broadcasting – Service Information
DVB-C	Digital Video Broadcasting – Cable
DVB-S	Digital Video Broadcasting - Satellite
DVB-S2	Digital Video Broadcasting – Satellite 2nd generation
DVB-T	Digital Video Broadcasting – Terrestrial
DVB-T2	Digital Video Broadcasting – Terrestrial 2nd generation
EPG	Electronic Programme Guide
FQDN	Fully Qualified Domain Name
FDP	File Delivery Protocol
HDMI	High Definition Multimedia Interface
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol - Secure
IP	Internet Protocol
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
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 TS 102 796 [1]. 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 this 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 TS 102 796 [1], 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 this 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.

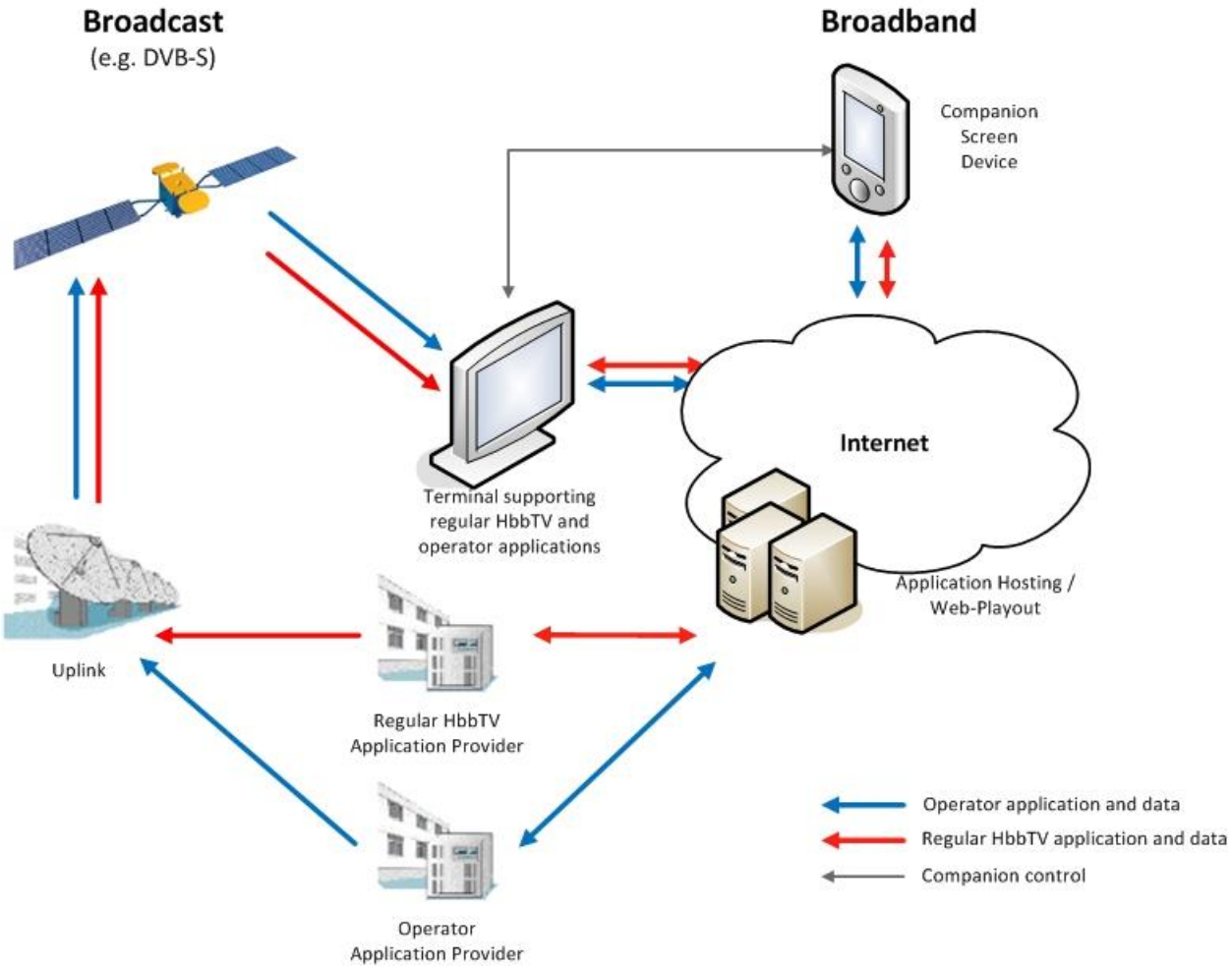


Figure 1: System overview for an operator application terminal

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.

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	Hide operator application

The behaviour of the EXIT button is defined in this document and implemented by the manufacturer. 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.

5.3.2 Start page

The bilateral agreement may define different start pages of the operator application 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 6 combinations of (XML) AIT discovery, download and appliciation 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
(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	Not included in the present document						
Operator application package download	Via IP	6.1.7.2	✓	✓	✓	✓	✓	
	Via DSM-CC object carousel	6.1.7.3						✓
	From CICAM auxiliary file system	Not included in the present document						

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 below.

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 6.5) 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 EN 300 468 [16] 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 TS 102 796 [1].

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 standardised 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 and 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 “https” 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.

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 FQDN is specified may define which countries are appropriate for its use.

6.1.3.7 DNS SRV lookup to a standardised 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 won't 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 RFC2782 [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.

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 RFC2782 [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 RFC2782 [6].

If the terminal discovers the location of one or more XML AIT 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 TS 102 796 [1] with mutual authentication as defined in clause 11.2.1 of the present document.

Terminal shall 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.

The terminal shall parse the XML AIT according to the requirements defined in clause 7.2.3.2 of TS 102 796 [1] with the additional requirements in Table 7 and clause 7.2.17.3.1 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.

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.

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 TS 102 796 [1].

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 TS 102 796 [1] with the additional requirements in table 8. 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` "

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.

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 two mechanisms by which an encrypted application package can be downloaded:

- HTTP over TLS via broadband IP and
- DSM-CC object carousel over broadcast.

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 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.

6.1.8 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).

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.

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 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’). This 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.

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.

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.

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 1)
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-foreground	Optional (see note 1)
Overlaid transient	Terminal UI appears on top of operator application zapping banner	No	overlaid-transient	Yes
NOTE 1: 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 TS 102 796 [1]).

If the user triggers the "EXIT or comparable button" mechanism as defined in clause 10.2.2.1 of TS 102 796 [1], this shall have no effect on an operator application except when it is in the foreground or transient state. 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

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 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 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: 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: 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 reserved keys and operator application keys, as defined by clause 10.1.3.
- The operator application has access to the regular 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 and the user triggers the "EXIT or comparable button" mechanism as defined in clause 10.2.2.1 of TS 102 796 [1].
- 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 .

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
- The operator application has access to the operator application reserved keys, as defined by clause 10.1.3.
- The operator application might have access to some of the regular 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.

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, overlaid foreground or background state and it makes a 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 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 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: 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: 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 reserved keys and operator application keys, as defined by clause 10.1.3
- The operator application has access to the regular 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 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 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 fail if none of the above conditions apply
- 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 reserved keys, as defined by clause 10.1.3
- The operator application may have access to the regular 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 foreground, overlaid foreground or background state and it makes a call to the `opAppRequestTransient` method.
  - The terminal shall accept the request if it was made within a handler for ChannelChangeSucceeded events in the operator application
  - The terminal shall accept the request if it was made within a handler for keydown, keyup or keypress events in the operator application
  - The terminal shall accept the request if it was made within a handler for click events for a notification requested by the operator application and activated by the user.
  - The terminal shall accept the request 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 terminal shall reject the request if none of the above conditions apply
- 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 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: 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 Reserved keys, as defined by clause 10.1.3
- The operator application may have access to the regular 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.

6.4 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 10.2.2) 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 10.2.2) 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 6.2.2.3.2.

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 TS 102 796 [1], 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 TS 102 796 [1].

- Background state: terminals shall act on this signalling as defined in TS 102 796 [1].

6.5.2 Starting and stopping regular HbbTV applications

The following requirements are in addition to those defined in clause 6.2 of TS 102 796 [1].

- 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 TS 102 796 [1], 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 TS 102 796 [1] which may require the terminal to stop or start a broadcast-related application.

- 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 TS 102 796 [1]).
    - 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.

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 reserved keys and 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 standardised 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 5.2.1) including starting them if they are not running.

NOTE: 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.

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 TS 102 796 [1] 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 it's 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).	(startup location not signalled)
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 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`

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 [18]. 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 TS 102 796 [1] 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	0xnn
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 0xnn 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 TS 102 796 [1] 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	0xnn
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 0xnn 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 TS 102 796 [1] 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 10 of an application with a minimum version of 8. The effect of this would be that once version 10 has been installed, any attempt to install versions 1 to 7 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>10</mhp:version>
                    <mhp:mhpVersion>
                        <mhp:profile>0</mhp:profile>
                        <mhp:versionMajor>1</mhp:versionMajor>
                        <mhp:versionMinor>3</mhp:versionMinor>
```

```
                        <mhp:versionMicro>1</mhp:versionMicro>
                    </mhp:mhpVersion>
                    <hbb:MinimumApplicationVersion>8</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/v10.pkg</mhp:applicationLocation>
            </mhp:Application>
        </mhp:ApplicationList>
    </mhp:ApplicationDiscovery>
</mhp:ServiceDiscovery>
```

7.4 Operator application ZIP File

7.4.1 Operator application ZIP File Format

The operator application ZIP File shall conform to the “ISO/IEC 21320-1 Document Container File” specification [10].

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	‘	allowed
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	`	allowed
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.

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

8 Browser application environment

8.1 Execution model

Clause 5.1 of the HTML5 specification (https://www.w3.org/TR/html5/browsers.html#windows indirectly referenced through TS 102 796 [1]) defines the concept of browsing contexts – "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 TS 102 796 [1]. 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

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

This clause defines a generic mechanism for terminals to provide access to 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".

The following functions shall be available on the global JavaScript context to support proprietary features of the terminal.

`String[] getTVProprietaryFunctions(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.	
Arguments	`namespace`	This parameter limits the returned function names to the specified namespace. If it is empty, all available proprietary functions shall be returned.

Boolean queryTVProprietaryFunction(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.

Object invokeTVProprietaryFunction(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.	
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.

Example usage of the proprietary TV function available in 'com.some_tv_manufacturer' namespace.

```
// Check if the dimTheDisplay function exists on the terminal.
if (queryTVProprietaryFunction('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 {
        invokeTVProprietaryFunction('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 getTVProprietaryFunctions('')) {
            throw Error('Function ' + name + ' is not available');
        } else {
            invokeTVProprietaryFunction('', 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 {
    oipfObjectFctory.dimTheDisplay(20, dimResultClbk);
} catch (err) {
```

```
console.log('The dimTheDisplay function could not be called');
}
```

8.4 Web APIs

8.4.1 Web Notifications

8.4.1.1 Requirements

The Web Notifications specification [8] shall be supported with the following additional requirements;

- By default the user agent shall grant permission to display these for all origins used by operator applications for which it has a bilateral agreement - see clause 4.3 of that specification. Hence the permission to display notifications for those origins shall be "granted".
- 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: 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: It is implementation dependent whether notifications are queued or discarded under these circumstances.

- All notifications originating from an operator applications shall be discarded when that applications 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: 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.

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 TS 102 796

8.5.1 Modification to terminalChannel

Clause 8.2.5 of TS 102 796 [1] adds a new readonly 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 TS 102 796 [1] and referenced specifications.
- Broadcast video presentation by the terminal itself is defined in TS 102 796 [1].
- 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 TS 102 796 [1] 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 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

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 TS 102 796 [1] 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 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 x 720 coordinate system defined in clause 10.2.1 of TS 102 796 [1].
- 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 TS 102 796 [1] 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.

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.

NOTE 2: Neither the present document nor TS 102 796 [1] 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 TS 102 796 [1] 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

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.

9.3 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:

- Operator applications shall be displayed using either the "Platform-specific application graphic plane" or the "DAE application graphic plane" as defined in that clause. Operator applications 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).

- Terminals shall support simultaneous display of 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.

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 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 TS 102 796 [1] 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.

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 TS 102 796 [1]. 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 TS 102 796 [1].

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]
- `preferredSubtitleLanguage`, as defined in clause 7.3.2 of the OIPF DAE specification [2]
- `audioDescriptionEnabled`, as defined in clause A.2.20 of TS 102 796 [1]
- `subtitlesEnabled`, as defined in clause A.2.20 of TS 102 796 [1]

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 TS 102 796 [1]) 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 BroadcastSupervisor object is destroyed:
  - 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 TS 102 796 [1] 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 TS 102 796 [1].

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 TS 102 796 [1] 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 TS 102 796 [1] 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

10.1.2 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 TS 102 796 [1]	Unchanged from TS 102 796 [1]	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 TS 102 796 [1]. 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 that specification 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 and operator application reserved 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 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 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 application keys that it has requested to receive using the KeySet API except for those regular 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 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 TS 102 796 [1], 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 reserved 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 reserved keys or the system keys listed in 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 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 TS 102 796 [1] or if the terminal has acted on a press of the "EXIT or comparable button" to terminate a regular HbbTV application or send an operator application into the background state.

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 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	
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.		

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 TS 102 796 [1].

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.

This specification 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 [1] 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 this specification.

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 TS 102 796 [1]. 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 this specification.

Operators shall use TLS server certificates compatible with the specifications detailed in clause 11.2 "TLS and Root Certificates" of TS 102 796 [1].

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.

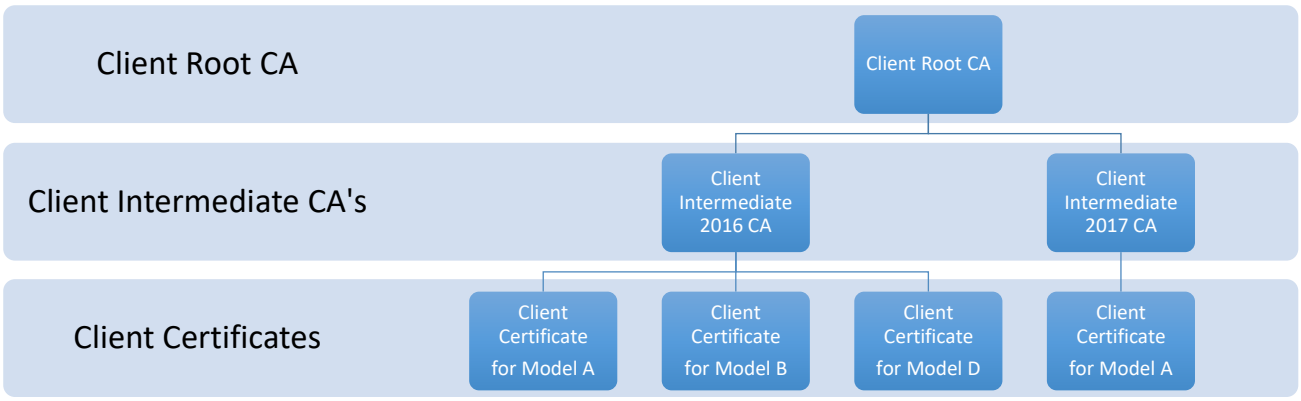


Figure 2: Example TLS Client Certificate Hierarchy

NOTE: 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 of inclusion on a terminal.

Negotiation and delivery of the TLS client certificate to the server is defined by the TLS Specification [12].

TLS client certificates shall comply with IETF RFC 5280 [13].

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 [13]

Operators should regulary (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 RFC 5280 [13] with the additional constraints profiled in Table 18 & 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 RFC 5280 [13] with the additional constraints detailed in Table 18 & 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 RFC 4055 [15].
`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 RFC 5280 [13] with the additional constraints profiled in Table 20 & 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 RFC 4055 [**15**].
`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 7.3.2.4 of TS 102 796 [1]. 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 TS 102 796 [1]. Any other mandatory attributes of the `subject` field defined in RFC 5280 [13] 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 of a trusted manufacturer. If the granularity of all the terminal packaging certificates is an individual model then the operator application will only run on trusted models. Under these circumstances, the HTTP user agent string can be considered trustworthy to distinguish between individual trusted models.

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 (RFC 5652 [14]) 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 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 either the Operator Signing Root CA certificate or one of the 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 this 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.

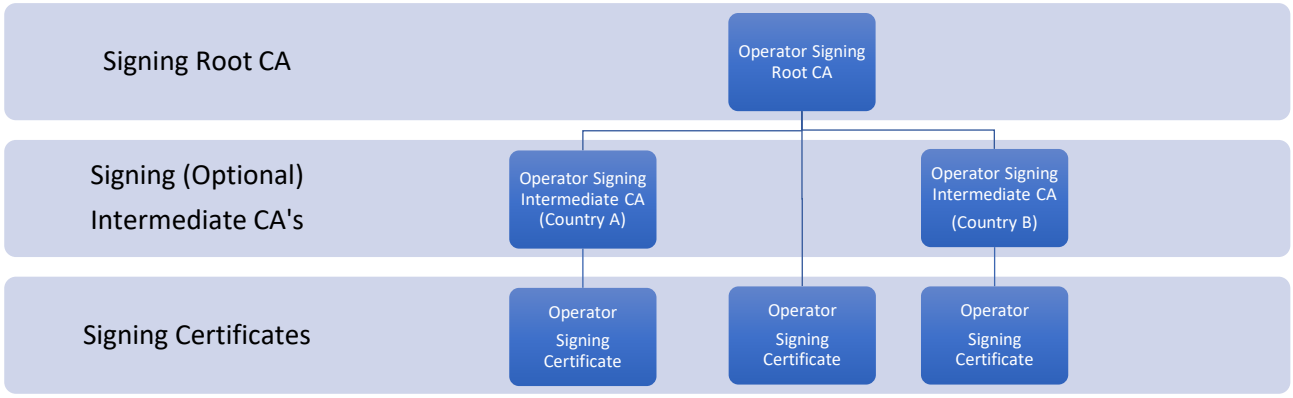


Figure 3: Example of an Operator Signing Root CA Hierarchy

The Operator Signing Certificate shall be a standard X.509 v3 certificate conforming to RFC 5280 [13] with the following additional constraints:

- The public key shall be an RSA key of length at least 2048 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 RFC 3447.

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 RFC 4055 [15].
`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 organization_id of the operator. The organization_id shall be encoded as an unsigned decimal integer without any leading zeros. Any other mandatory attributes of the `subject` field defined in RFC 5280 [13] 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 this 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.

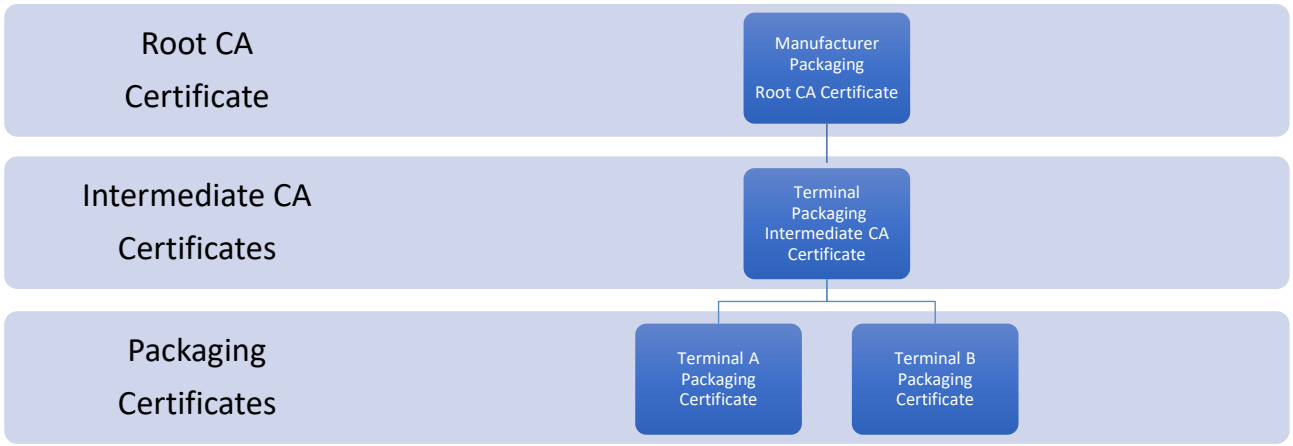


Figure 4: Example of a Manufacturer Packaging Certificate Hierarchy

The Terminal Packaging Certificate shall be a standard X.509 v3 certificate conforming to RFC 5280 [13] 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 RFC 3447.

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 RFC 4055 [15].
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 TS 102 796 [1]. 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 TS 102 796 [1]. Any other mandatory attributes of the subject field defined in RFC 5280 [13] 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 RFC 5652 [14], 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 RFC 5652 [14] 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 RFC 5652 [14], 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 RFC 5652 [3] using a ktri field of type KeyTransRecipientInfo as described in section 6.2.1 of RFC 5652 [14].
- 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 RFC 5652 [14].

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 RFC 5652 [14]. 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 RFC 5652 [14].

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 is invalid or any of the certificates have expired.

- The Operator Name, as signalled via the Organization ('O=') attribute of the `subject` field, or the organization_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.

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 [17] 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.

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 or persistent storage capabilities are introduced for use by operator applications. Only those capabilities in TS 102 796 [1] are provided, along with the privacy controls defined for them.

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 synchronisation functions as regular HbbTV applications. Operator applications shall be able to enable multi-stream and inter-device synchronisation 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 synchronisation 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 TS 102 796 [1] 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 TS 102 796 [1] 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 TS 102 796 [1] to avoid collisions with regular HbbTV applications also using this feature.

Annex A (normative): OIPF specification profile

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

Table A.1 below defines clauses of the OIPF DAE specification [2] that are required by the present document but which are either not required at all by TS 102 796 [1] or where the present document introduces additional requirements beyond what is required by that specification. Where a class or object is partly required by TS 102 796 [1], the properties and / or methods and / or events required by that specification are required by the present document. Only additional requirements are listed here. Methods properties and events that are not required by TS 102 796 [1] 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 [1].

Section, sub-section	Reference in DAE [2]	Status in HbbTV	Terminals supporting privileged operator applications	Terminals supporting operator-specific operator applications
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 [DAE]. 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 6.2.2.3. 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 TS 102 796 [1] 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.	
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 TS 102 796 [1] 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.	The requirements for privileged operator applications apply. Support for read and write access to the following properties is mandatory: • `preferredUILanguage` • `countryId`

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 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 following properties and methods are mandatory if the terminal supports operator applications controlling the power state: • `powerState` • `previousPowerState` • `timeCurrentPowerState` • `onPowerStateChange` • `setPowerState`
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.

Section, sub-section	Reference in DAE [2]	Status in HbbTV	Terminals supporting privileged operator applications	Terminals supporting operator-specific operator applications
The application/oipfParentalControlManager embedded object	7.9.1	M(*)	The following properties and methods are mandatory: • `isPINEntryLocked` • `setParentalControlStatus` • `getParentalControlStatus` • `unlockWithParentalControlPIN` In addition, the extensions in A.2.8 shall be supported. If the target argument of the `unlockWithParentalControlPIN` method is an HTML5 media element, the content being presented through this object shall be unlocked until a new item of content is played using this object.	The requirements for privileged operator applications apply. The following methods are mandatory: • `verifyParentalControlPIN` • `setParentalControlPIN`
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 TS 102 796 [1] with the modifications in clause A.2.7.	
The ScheduledRecording class	7.10.2	M-P(*)	The following properties are mandatory: • `state` • `error`	
Extension to application/oipfRecordingScheduler for control of recordings	7.10.4	M-P(*)	The following properties and methods are mandatory: • `onPVREvent` • `stop` • `update` The `PVREvent` class shall be supported.	
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`
video/broadcast embedded object	7.13.1	M(*)	The modifications in A.2.4 shall be supported	The modifications in 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	Unchanged	Unchanged
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

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 creating Channel lists from SD&S fragments	7.13.8	NI	Unchanged	Unchanged
ChannelConfig class	7.13.9	M(*)	The `getBroadcastSupervisor` method shall be supported (see clause A.2.3).	The requirements for privileged operator applications apply. The `setChannelList` and `createChannelList` methods (see 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` • `createChannelScanParametersObject` • `createChannelScanOptionsObject` • `onChannelScan` • `onChannelListUpdate` • `ChannelScan event` • `ChannelListUpdate event`
ChannelList class	7.13.10	M(*)	Unchanged	Unchanged
Channel class	7.13.11	M(*)	Unchanged	The extensions in 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
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

Section, sub-section	Reference in DAE [2]	Status in HbbTV	Terminals supporting privileged operator applications	Terminals supporting operator-specific operator applications
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 DVB service 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)

			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)	
UI_MESSAGES_REMINDER	33	B	Used to request suppression of scheduled messages (see section 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 (e.g. "Signal lost" or "Upgrade available")	
	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 5.8.2). 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)*	

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 reserved 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 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.

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 TS 102 809 [3].

`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.	
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 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 the following table.

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 1)
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 1: 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 this 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 this 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.

`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_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. The terminal shall only grant the request if it was received in accordance with the requirements listed in clause 6.3.3.4 of the present document. If the request is granted then: • the Application shall be notified via an event dispatched to the `onOperatorApplicationStateChange` function and • the terminal shall start a timer with a duration of 1 minute and • the method shall return `true`. 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` function. 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` function 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` function.

`void opAppRequestUpdate(Boolean immediate)`		
Description	When called by an operator application, this requests the terminal to update that operator application. The process of updating the app is asynchronous.	
Arguments	immediate	If true, indicates whether the update shall happen immediately. If false, indicates that the update should happen 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.

`Integer opAppUpdateStatus()`

<table>
<tr><td>Description</td><td>Returns the current status of any ongoing activity to update this operator application. The value returned by this function shall be:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>-2</td><td>No update is in progress.</td></tr>
<tr><td>-1</td><td>New software is available to download for the operator application but download has not yet started.</td></tr>
<tr><td>0 … 99</td><td>New software for the operator application is being downloaded to the terminal and the value gives an approximation of the amount already downloaded.</td></tr>
<tr><td>100</td><td>Indicates that new software for the operator application has been successfully downloaded to the terminal and is about to be installed.</td></tr>
<tr><td>1001 … 1999</td><td>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.</td></tr>
<tr><td>2000 - 2999</td><td>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.</td></tr>
</table></td></tr>
</table>

<table>
<tr><td colspan="3"><code>Application createApplication(String uri, Boolean createChild, Boolean runAsOpApp)</code></td></tr>
<tr><td>Description</td><td colspan="2">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 <code>null</code>. Behaviour if the application can be created but subsequently fails to load shall be as defined by clause 6.2.2.5.6 of TS 102 796 [1].

If a regular HbbTV application calls this method with runAsOpApp being <code>true</code> or createChild being <code>true</code> then the method shall return null. If a regular HbbTV application calls this method with runAsOpApp being <code>false</code> and createChild being <code>false</code> then this shall be identical to calling the method without either argument.

Launching a new application to run as an operator application (runAsOpApp being <code>true</code>) shall replace the calling operator application with a new operator application in the same way as a regular HbbTV application calling <code>createApplication(String uri, false)</code> 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. 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 <code>false</code>) 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 TS 102 796 [1]. 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 <code>ApplicationLoaded</code> event when the launched application loads successfully and with an <code>ApplicationUnloaded</code> event when it exits</td></tr>
<tr><td rowspan="3">Arguments</td><td>uri</td><td>The URI of the application to launch.</td></tr>
<tr><td>createChild</td><td>When runAsOpApp is false, defines whether the launching application is to be notified when the launched application exits (<code>true</code>) or not (<code>false</code>). This shall be ignored when runAsOpApp is true.</td></tr>
<tr><td>runAsOpApp</td><td>This optional argument defines whether the application shall be launched as an operator application (<code>true</code>) or a broadcast-independent regular HbbTV application (<code>false</code>). If the argument is not included then the value shall be considered to be false.</td></tr>
</table>

<table>
<tr><td><code>Boolean opAppUninstall()</code></td></tr>
</table>

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 applicaton 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.

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, reason`
`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. 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.

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.

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 TS 102 796 [1].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. 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.
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 • the equivalent state of the terminal software if the terminal is presenting broadcast content to the user.
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.
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.
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 [] 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.

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)	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).
prevChannel	Shall work as specified for a video/broadcast object except 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).
nextChannel	Shall work as specified for a video/broadcast object except 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).
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
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

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 a 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 method `getScheduledRecordings`() 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 method `getInProgressRecordings`() shall return all recordings that are currently in progress regardless of the origin of the application that scheduled them.

- For operator-specific operator applications (and not privileged operator applications), the following shall apply:
  - The method `remove`() 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 method `remove`() 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.

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

EN 300 468 [16] 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 Annex B.1.

Table Annex 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. 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
0x1-0xF	Reserved for future use

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.

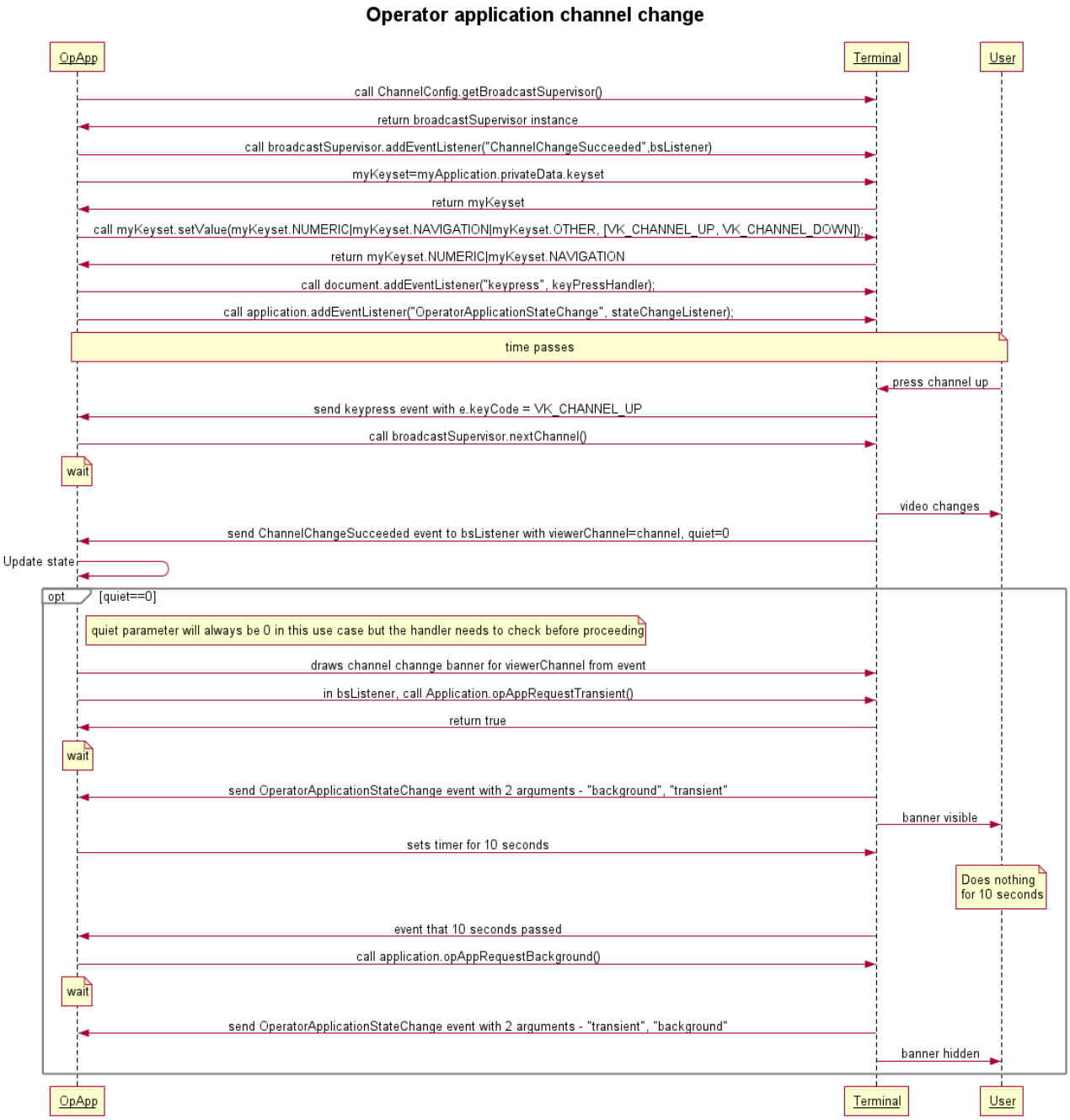
min_polling_interval: This 16-bit field is specified in ETSI EN 300 468 [16] in the definition of the URI_linkage_descriptor.

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 below 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.



C.2 Broadband-based operator application discovery

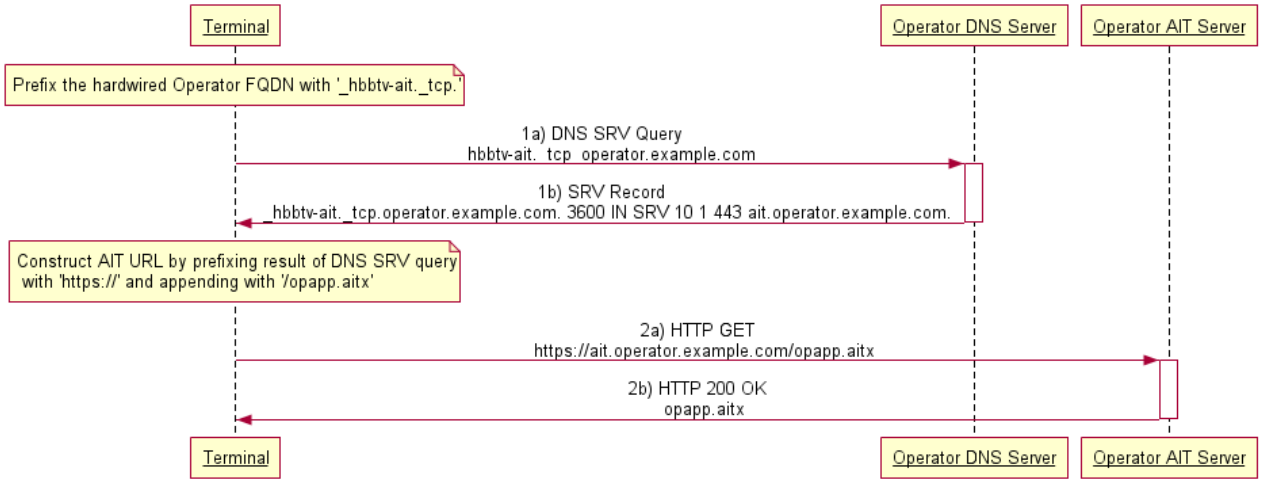
The following sequence diagrams shows an example of broadband-based operator application discovery.

The complete flow diagram is shown in figure [annex C.1]. In this example 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 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”.

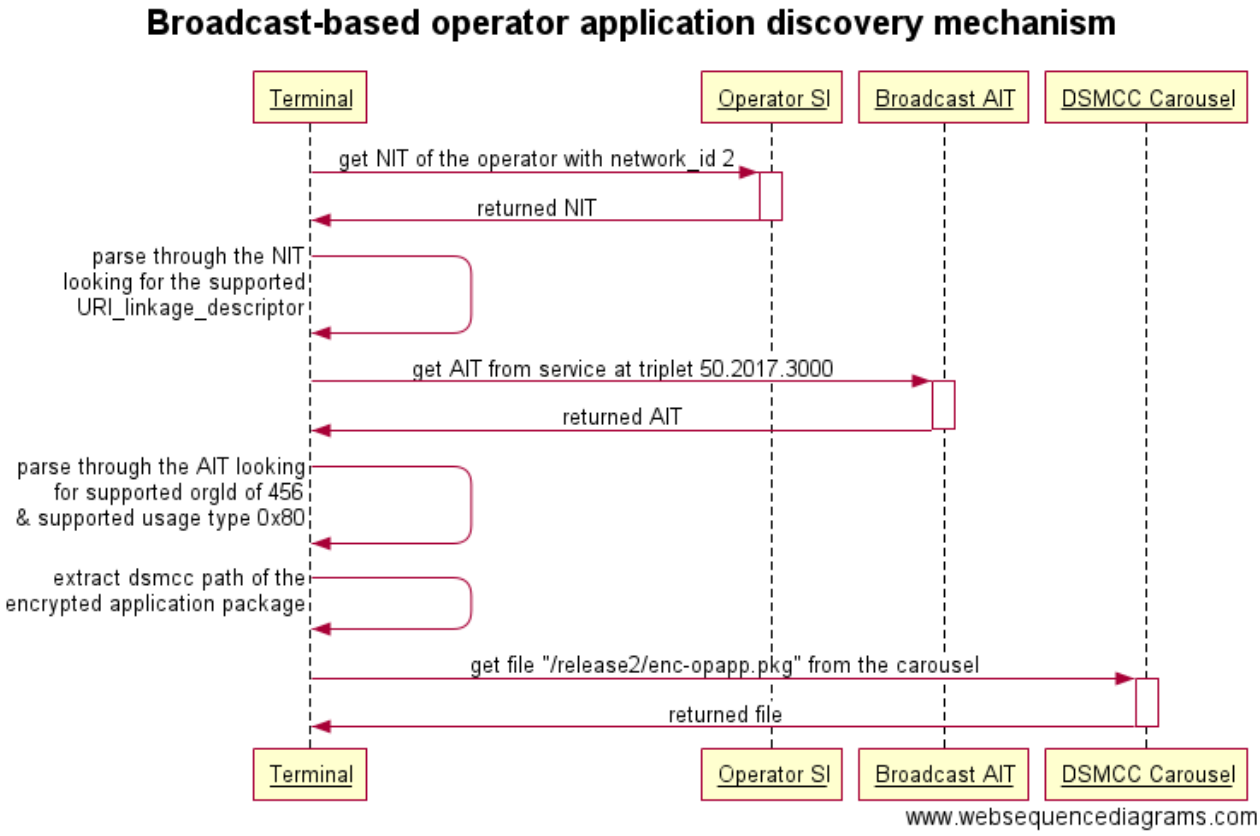
Using the AIT server name the HbbTV terminal can construct now an HTTPS GET request by prefixing "https://" and appending "/opapp_ait.xml" to the SRV response. In Step 2a) the constructed GET command is sent to the identified AIT server and this server responds to the request in Step 2b) with the 200 OK message and the wanted XML file.

Figure C.5: Operator Application Discovery from Hardwired FQDN



C.3 Broadcast-based operator application discovery and installation

The example in figure X shows an operator with an operator application having an organisation 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 DSMCC 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.



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	Ref.	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.
Non-connected operation		The bilateral agreement may define if the operator application needs an operational broadband connection.
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. • This may include the definition of different launch context signalling.
Power-on behaviour	6.3.2.1	The bilateral agreement may define the power-on behaviour. • This may include the automatic restart of the operator application either after all reboots or after reboots where the operator application was running previously.
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	The bilateral agreement may define sizes of icons that are provided by the operator (see Table 7: Definition of different icon flags of TS 102 809).
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 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 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.
Mandatory API	8.2	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.
Web notifications	8.4.1	By default the user agent shall grant permission to display these for all origins used by operator applications for which it has a bilateral agreement - see clause 4.3 of that specification.
Operator application state change	A.2.2.1	The reason argument shall be an appropriate value from either table 10 or table 11 or some other value as defined in the bilateral agreement that indicates the reason the operator application has changed it's state.
Error codes	A.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	11.2.2	The bilateral agreement needs to address how the manufacturer provides their Client Root CA Certificate (or an intermediate) to the operator.
Scope of operator application		
Replacement of UI elements	6.4	The bilateral agreement needs to define which UI elements of the terminal the operator application replaces. • This may include the definition of additional (user defined) constants to be used with the `replaceUIElements` method.
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	The bilateral agreement may define whether the operator application or the terminal implements the regulatory requirements for parental control.
Channels outside the set of channels aggregated by the operator	4.1.7 5.3.2	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.
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
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.

Annex (informative): Bibliography

-

Annex (informative): Change History

Date	Version	Information about changes
<Month year>	<#>	<Changes made are listed in this cell>

History

Document history		
<Version>	<Date>	<Milestone>

Latest changes made on 2016-05-20